TUGAS AKHIR – KI141502

PENERAPAN ALGORITMA ALPHA BETA PRUNING SEBAGAI KECERDASAN BUATAN PADA GAME PAWN BATTLE

IROOYAN ALFI AZIZ T.S
NRP 5112100089

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.
Isye Arieshanti , S.Kom., M.Phil.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

FINAL PROJECT– KI141502

IMPLEMENTATION OF ALPHA BETA PRUNING ALGORITHM FOR ARTIFICIAL INTELLIGENCE IN PAWN BATTLE GAME

IROOYAN ALFI AZIZ T.S
NRP 5112100089

Advisor
Imam Kuswardayan, S.Kom., M.T.
Isye Arieshanti, S.Kom., M.Phil.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2015

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji dan syukur, kehadirat Allah Subhanahu wa ta'ala yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "PENERAPAN ALGORITMA ALPHA BETA PRUNING SEBAGAI KECERDASAN BUATAN PADA GAME PAWN BATTLE".

Pengerjaan tugas akhir ini adalah momen bagi penulis untuk mengeluarkan seluruh kemampuan, hasrat, dan keinginan yang terpendam di dalam hati mulai dari masuk kuliah hingga lulus sekarang ini, lebih tepatnya di kampus Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan-bantuan yang penulis terima dari berbagai pihak. Melalui lembar ini, penulis ingin secara khusus menyampaikan ucapan terima kasih kepada:

1. Allah SWT atas limpahan rahmat dan rezeki-Nya sehingga penulis dapat menyelesaikan Tugas Akhir.
2. Ayah penulis, Kateno, dan Ibu penulis, Siti Yatimah, yang selalu memberikan dukungan, doa, perhatian, dan kasih saying yang tidak terhingga.
3. Teman-teman penulis, Arika Saputro, Imaduddin Al Fikri dan Mas Dika, yang selalu memberikan motivasi dan bimbingan selama menyelesaikan studi penulis.
4. Febriandini Ayu, yang telah membantu dan senantiasa memberi semangat untuk menyelesaikan studi penulis.
5. Bapak Imam Kuswadaryan selaku dosen pembimbing Tugas Akhir pertama dan yang telah memberikan arahan dalam pengerjaan Tugas Akhir ini.
6. Ibu Isye Arieshanti selaku dosen pembimbing Tugas Akhir kedua yang dengan sabar membimbing penulis dalam pengerjaan Tugas Akhir ini.

7. Bapak Ridho Rahman selaku dosen wali yang berkenan memberi saran dan arahan selama penulis menjalani studi S1.
8. Bapak Radityo Anggoro selaku dosen koordinator Tugas Akhir yang telah membantu penulis atas segala sesuatu mengenai syarat-syarat dan terlaksananya sidang Tugas Akhir.
9. Bapak Darlis Herumurti selaku Ketua Jurusan Teknik Informatika ITS yang selama ini memberikan bantuan kepada penulis.
10. Dosen-dosen Teknik Informatika yang dengan sabar mendidik dan memberikan pengalaman baru kepada penulis selama di Teknik Informatika.
11. Staf TU Teknik Informatika ITS yang senantiasa memudahkan segala urusan penulis di jurusan.
12. Rekan-rekan dan pengelola Laboratorium Interaksi, Grafik, dan Seni yang telah memberikan fasilitas dan kesempatan melakukan riset atas Tugas Akhir yang dikerjakan penulis.
13. Rekan-rekan dan sahabat-sahabat penulis angkatan 2012 yang memberikan dorongan motivasi dan bantuan kepada penulis.
14. Pihak-pihak lain yang tidak sengaja terlewat dan tidak dapat penulis sebutkan satu per satu.

Penulis telah berusaha sebaik mungkin dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Januari 2015
Penulis

Irooyan Alfi Aziz T.S

LEMBAR PENGESAHAN

PENERAPAN ALGORITMA ALPHA BETA PRUNING SEBAGAI KECERDASAN BUATAN PADA GAME PAWN BATTLE

Tugas Akhir

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Interaksi, Grafika, dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
IROOYAN ALFI AZIZ T.S
NRP. 5112 100 089

Disetujui oleh Dosen Pembimbing Tugas Akhir:
Imam Kuswadaryan, S.Kom., M.T. .....
NIP: 19761215 200312 1 001 (pembimbing 1)

Isye Arieshanti, S.Kom., M.Phil. .....
NIP: 19780412 200604 2 001 (pembimbing 2)

SURABAYA
DESEMBER, 2015

PENERAPAN ALGORITMA ALPHA BETA PRUNING SEBAGAI KECERDASAN BUATAN PADA GAME PAWN BATTLE

Nama Mahasiswa : Irooyan Alfi Aziz T.S
NRP : 5112 100 089
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Imam Kuswadaryan, S.Kom., M.T.
Dosen Pembimbing II : Isye Arieshanti, S.Kom., M.Phil.

ABSTRAK

Saat ini game memiliki perkembangan yang sangat pesat terutama game-game yang berjalan dalam platform desktop. Banyak dari game-game tersebut hanya memprioritaskan unsur adiktif dan kurang memperhatikan unsur edukatif di dalam game.

Namun, seperti yang kita tahu, permainan catur adalah permainan kuno yang membantu kita mengasah kemampuan kognitif dan strategi berpikir. Dengan menggabungkan unsur adiktif dalam game modern dan membangun strategi berpikir dalam catur, dalam Tugas Akhir ini dibangun permainan yang membantu pemain mengasah strategi berpikir dengan menyenangkan.

Tugas Akhir ini menerapkan role-play pawn battle yang tidak jauh berbeda dengan catur. Pawn battle adalah permainan dimana pemain hanya memainkan pion dengan peraturan-peraturan resmi permainan catur. Aplikasi ini menerapkan algoritma alpha beta pruning sebagai kecerdasan buatan dengan level yang bisa diatur sehingga memudahkan pemain memilih lawan bermain. Sehingga diharapkan pemain yang masih dalam tahap pemula maupun sudah mahir bisa bermain di levelnya masing-masing dengan nyaman.

Kata kunci**: **Alpha Beta Pruning, Kecerdasan Buatan, Catur, Pawn Battle.

IMPLEMENTATION OF ALPHA BETA PRUNING ALGORITHM FOR ARTIFICIAL INTELLIGENCE IN PAWN BATTLE GAME

Student Name	: Irooyan Alfi Aziz T.S
NRP	: 5112 100 089
Major	: Teknik Informatika FTIf-ITS
Advisor I	: Imam Kuswadaryan, S.Kom., M.T.
Advisor II	: Isye Arieshanti, S.Kom., M.Phil.

ABSTRACT

Today, games have developed very rapidly, especially games that tun on desktop platform. Many of them just concerned on addictive elements and less-concerned in educative elements of the game..

But, as we know, chess board game is 'old but gold' games that help us sharpen our strategic thinking and cognitive ability. By combining addictive elements in modern game and strategic developing elements in chess game, in this final project is built an enjoyful game that helps user developing their strategic thinking.

This final project is using pawn battle role-play that not so different with old chess board game. Pawn battle is game where the user just using their pawn with officially chess rules for winning the game. This application using alpha beta priuning algorithm as artificial intelligence with easy-to-set level changing. So, the user could easily choose their level AI opponents. Therefore, user should play this game delightfully.

Keywords: Alpha Beta Pruning, Artificial Intelligence, Chess, Pawn Battle.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Bab ini memaparkan garis besar Tugas Akhir yang meliputi latar belakang, tujuan dan manfaat pembuatan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Permainan catur adalah permainan kuno yang hingga saat ini masih populer dimainkan. Permainan ini berkembang pesat, tidak hanya dimainkan oleh orang-orang dewasa saja, anak kecil bahkan orang tua pun banyak yang memainkan permainan ini. Permainan catur banyak diminati karena mampu mengasah pemikiran kognitif dan tidak perlu membuang banyak tenaga seperti olahraga lain pada umumnya.

Saat ini berkembang luas strategi-strategi dalam bermain catur. Apabila seorang pemula hanya diberikan strategi-strategi tinggal pakai, maka akan membuat seorang pemula tersebut tidak berkembang dan hanya mengandalkan hafalan. Sehingga apabila pemula tersebut dihadapkan pada kasus-kasus khusus yang unik dan tidak sesuai dengan strategi yang telah diajarkan sebelumnya, akan membuat pemula tersebut bermain buruk.

Oleh karena itu, dibutuhkan tidak hanya strategi-strategi matang siap pakai, tetapi juga latihan-latihan dasar permainan catur. Pawn Battle adalah penyederhanaan dari permainan catur. Permainan ini jauh lebih mudah dimainkan daripada catur, sehingga pemula tidak menghabiskan energi berpikir seperti catur. Pawn Battle berguna melatih seorang pemula untuk bisa berpikir efektif dan belajar untuk menggunakan bidak pion selagi berlajar strategi dasar yang sangat penting dalam bermain catur.

Umumnya, algoritma kecerdasan buatan dalam permainan catur adalah minimax. Kemudian dikembangkan menjadi alpha beta pruning. Algoritma minimax menemukan langkah terbaik dan algoritma alpha beta pruning mencegah untuk meng- expand cabang

/ node yang tidak dapat menghasilkan hasil yang lebih baik daripada cabang sebelumnya yang sudah disimpan.

Namun, metode yang saat ini berkembang ialah dengan mengevaluasi posisi yang beracuan pada heuristic tertentu. Lebih lanjut program catur yang benar-benar dibuat untuk industri akan memiliki pustaka tertentu sehingga permainan dimulai hanya dengan pustaka-pustaka tersebut dan tidak terus menerus memeriksa node / cabang permainan.

Karena metode yang umum digunakan dan dikenal cukup efektif, serta program yang akan dibuat hanya meliputi pion yang kemungkinan langkahnya tidak terlalu banyak seperti catur, maka algoritma yang akan digunakan untuk kecerdasan pada permainan Pawn Battle adalah alpha beta pruning. Serta game berbasis desktop ini akan berjalan pada sistem operasi Windows.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana merancang rule-based pada permainan *Pawn Battle*
2. Bagaimana mengimplementasikan rule-based yang telah dirancang dengan menerapkan *alpha beta pruning* sebagai kecerdasan buatan.

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Aturan permainan yang dipakai dalam *game* ini sesuai dengan aturan permainan catur pada umumnya
2. Permainan dimulai dengan pemain melangkah terlebih dahulu

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah untuk membuat game Pawn Battle dan mengimplementasikan algoritma Alpha Beta Pruning dalam kecerdasan buatan yang membantu pemain dalam memainkan game ini. Serta meningkatkan memberikan

latihan dan strategi dasar yang penting dalam permainan catur sehingga melatih pemikiran yang strategis terhadap pengguna..

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini antara lain:

1. Mendapatkan pengetahuan dan wawasan pada permainan *Pawn Battle*.
2. Meningkatkan pemikiran strategis dan cermat kepada pengguna.
3. Memberikan media hiburan bagi para pengguna.

1.6 Metodologi

Pembuatan tugas akhir dilakukan menggunakan metodologi sebagai berikut:

A. Studi literatur

Tahap Studi Literatur merupakan tahap pembelajaran dan pengumpulan informasi yang digunakan untuk mengimplementasikan Tugas Akhir. Tahap ini diawali dengan pengumpulan literatur, diskusi, eksplorasi teknologi dan pustaka, serta pemahaman dasar teori yang digunakan pada topik tugas akhir. Literatur-literatur yang dimaksud disebutkan sebagai berikut:

1. Algoritma *Alpha Beta Pruning*;
2. *Unity 3d Platforms*;
3. *Blender*;
4. *Artificial Intelligence : A Modern Approach*;

B. Perancangan perangkat lunak

Pada tahap ini dilakukan analisa awal dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang sedang dihadapi. Selanjutnya, dirumuskan rancangan sistem yang dapat memberi solusi terhadap permasalahan tersebut. Langkah yang akan digunakan pada tahap ini adalah sebagai berikut:

1. Memepelajari lebih dalam mengenai game *Pawn Battle*.
2. Perancangan sistem dan mekanisme *game Pawn Battle*.
3. Analisis kebutuhan non fungsional.

4. Analisis algoritma *Alpha Beta Pruning* dalam membangun *game Pawn Battle.*
5. Perancangan kecerdasan buatan pada *game Pawn Battle*.

C. Implementasi dan pembuatan sistem
Tahap implementasi merupakan tahap untuk membangun aplikasi permainan beserta sistem yang terkait. Aplikasi ini akan dibangun dengan bahasa pemrograman C# dengan menggunakan platform *Unity 3d*. Aplikasi yang akan dibangun berbasis perangkat desktop dan berjalan di sistem operasi *Windows.*

D. Uji coba dan evaluasi
Pada tahap ini akan dilakukan pengujian usabilitas terhadap perangkat lunak menggunakan data atau skenario yang telah dipersiapkan sebelumnya yakni dengan cara melakukan survei ke pengguna yaitu beberapa pengguna yang suka bermain *game* di sekitar lingkungan Teknik Informatika ITS. Survei dilakukan untuk mengukur tingkat kegunaan dari aplikasi yang dibuat dalam membantu pengguna.

E. Penyusunan laporan tugas akhir
Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan tugas akhir.

1.7 Sistematika Penulisan

Buku tugas akhri ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut.

BAB I PENDAHULUAN

Bab ini berisi latar belakang masalah, rumusan dan batasan permasalahan, tujuan dan manfaat pembuatan tugas akhir, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II TINJAUAN PUSTAKA

Bab ini membahas dasar pembuatan dan beberapa teori penunjang yang berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir ini.

BAB III ANALISIS DAN PERANCANGAN

Bab ini membahas analisis dari sistem yang dibuat meliputi analisis permasalahan, deskripsi umum perangkat lunak, spesifikasi kebutuhan, dan identifikasi pengguna. Kemudian membahas rancangan dari sistem yang dibuat meliputi rancangan skenario kasus penggunaan, arsitektur, data, dan antarmuka.

BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi algoritma *Alpha Beta Pruning*, dan antarmuka permainan.

BAB V PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dari aplikasi yang dibuat dengan melihat keluaran yang dihasilkan oleh aplikasi dan evaluasi untuk mengetahui kemampuan aplikasi.

BAB VI PENUTUP

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran untuk pengembangan aplikasi selanjutnya.

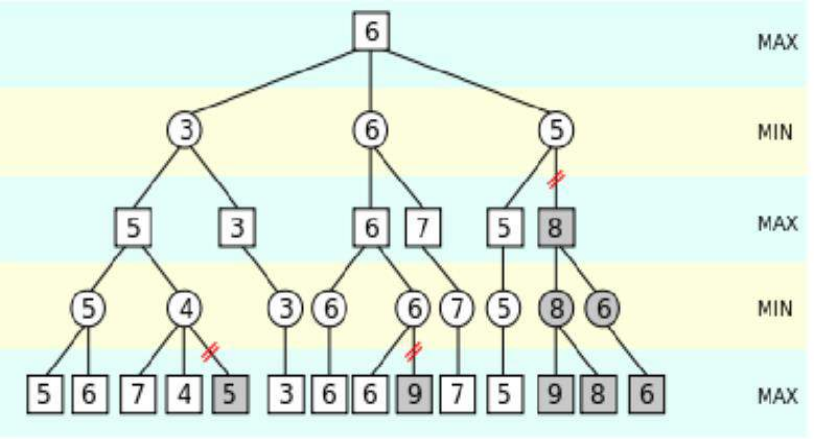
BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir. Teori-teori tersebut adalah Algoritma *Alpha Beta Pruning*, Permainan *Pawn Battle*, *Unity* dan *Blender*.

2.1 Algoritma *Alpha Beta Pruning*

Algoritma alpha beta pruning merupakan algoritma pencarian yang berusaha untuk mengurangi jumlah node yang dievaluasi oleh algoritma minimax dalam pohon pencarian. Algoritma ini umumnya dipakai untuk permainan dengan 2 pemain. Contohnya Tic-Tac-Toe, Othello, GO, atau catur. Pencarian akan berhenti ketika menemukan satu langkah terbaik dan langkah berikutnya terbukti lebih buruk dari langkah tersebut.

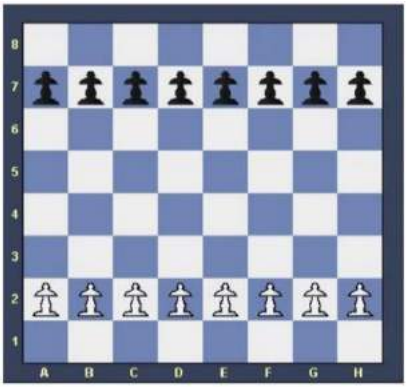
Algoritma Alpha beta pruning memiliki dua nilai yakni alpha dan beta. Alpha merupakan nilai maksimum yang dimiliki pemain dan beta merupakan nilai minimum yang dimiliki pemain. Awalnya, alpha ditetapkan tak terhingga negatif dan beta ditetapkan tak terhingga positif. Setiap pemilihan node, nilai alpha dan beta akan selalu diperbarui, hingga apabila terjadi nilai beta <= alpha, maka parent node tidak akan memilih node ini karena hanya akan membuat nilai parent node menjadi buruk.Akhirnya, cabang dibawah node tidak perlu dieksplorasi atau dengan kata lain dilakukan prune.



Gambar 2.1 Ilustrasi Algoritma *Alpha beta pruning*

2.2 Permainan *Pawn Battle*

Game ini hampir sama dengan permainan catur pada umumnya. Bedanya, jumlah bidak yang dimainkan hanyalah 8 buah per pemain. 8 buah bidak tersebut adalah bidak pion yang terletak pada baris ke-2 dan ke-8 bidang catur seperti pada gambar.



Gambar 2.2 Posisi awal bidak permainan *Battle Pawn*

Seperti dijelaskan dalam "Pawn Battle Rules and Strategies"[4] peraturan bermain Pawn Battle adalah:

1. Siapkan bidak pion pada baris ke-2 dan ke-8 seperti pada gambar 1. Kemudian pemain dengan bidak putih bermain lebih dahulu. Jalankan 1 pion sesuai dengan aturan yang akan dijelaskan dibawah nanti.

2. Pemain yang berhasil membawa pion ke sisi seberang atau berhasil memakan semua pion lawan adalah pemenangnya. Pemain juga dikatakan menjadi pemenang, apabila lawan menyerah.

3. Dikatakan imbang/seri apabila kedua pemain masih memiliki pion, namun tidak bisa berjalan (karena pion-pion yang tersisa terhalang pion lain). Sehingga terjadi kondisi "Stalemate".

4. Pemain hanya boleh menggerakkan 1 pion setiap gilirannya. Pemain tidak boleh melewati giliran (tidak menjalankan pion padahal masih memiliki langkah yang legal) atau menjalankan pion 2 kali.

Sedangkan menurut paper yang sama [4] aturan dasar menggerakkan pion adalah:

1. Pion hanya boleh bergerak maju, tidak boleh bergerak mundur atau kesamping. Jika ada pion lain yang menghalangi jalannya (berada di depannya), maka pion tersebut tidak bisa bergerak dan tidak bisa maju. Pion bisa memakan pion lawan yang berada di kotak diagonal depannya, kemudian posisi pion tersebut menggantikan posisi pion lawan yang dimakan. Contohnya, pion pada posisi c4 bisa memakan pion lawan yang berada di posisi d5 atau f5.

2. Umumnya, pion hanya bisa bergerak 1 langkah kedepan. Namun, pada langkah awal (ketika menempati posisi baris ke-2 atau ke-7), pion mampu memilih bergerak 1 kotak atau 2 kotak kedepan.

3. Jangan lupa peraturan en passant. Jika ada pion yang bergerak 2 kotak kedepan pada awal langkah (bergerak dari baris ke-2 menuju baris ke-4 atau dari baris ke-7 menuju baris ke-5) dan melewati pion musuh yang sudah lebih dulu berada di baris tersebut (baris 4 atau baris 5), maka pion lawan berhak memakan pion en passant pemain tersebut layaknya jika pion pemain tersebut berjalan

1 kotak. Aturan en passant ini berlaku hanya pada giliran selanjutnya. Apabila pada giliran selanjutnya, lawan tidak menggunakan haknya, maka aturan en passant ini tidak berlaku pada giliran berikutnya. Contohnya, pion hitam berada pada e4, kemudian pion putih bergerak dari d2 ke d4. Maka pion hitam bisa memakan pion tersebut dan posisinya berpindah ke d3.

4. Ketika ada pion yang mampu mencapai sisi seberang papan, maka pemain tersebut dinyatakan menang.

2.3 Unity

Unity adalah platform pengembang perangkat lunak yang fleksibel dan ampuh untuk membuat multiplatform permainan 3 dimensi dan 2 dimensi. Unity adalah ekosistem yang lengkap bagi siapapun yang hendak membangun sebuah bisnis dalam pembuatan konten high-end dan menghubungkan pemain-pemain dan pengguna mereka yang antusias dan loyal.

Unity adalah kakas yang berpengalaman di bidang memoles dan mengembangkan platform end-to-end. Bahkan, secara yakin, Unity memberikan target pada platform-platform yang sedang hangat saat ini, termasuk platform-platform pendatang seperti WebGL and Oculus Rift. Unity seceara efektif mengoptimasi penampilan dengan kakas platform silang dan melakukan deploy dengan hanya satu klik.

Adapun sistem operasi yang mendukung dan bisa memainkan produk-produk Unity adalah ios, android, windows phone, bb 10, tizen, windows, windows store apps, mac, linux, web player, WebGL, Play Station 3, Play Station 4 and morpheus, PsVita, Xbox one, Xbox 360, Wii U dan Oculus Rift

2.4 *Blender*

Blender adalah aplikasi gratis untuk membuat desain 3D. Blender mendukung keseluruhan pembuatan objek 3D mulai dari modelling, rigging, animasi, simulasi, rendering, compositing dan pelacakan gerakan. Bahkan, Blender juga mendukung video editing

dan penciptaan permainan. Untuk pengguna mahir, dapat menggunakan Blender

API yang sudah disediakan untuk scripting dengan bahasa Python, Sehingga pengguna tersebut dapat menambahkan aplikasi dan alat-alat khusus.

Blender adalah aplikasi cross-platform yang sama baiknya berjalan pada Linux, Windows, maupun Macintosh. Antarmuka blender menggunakan OpenGL dan untuk meyakinkan spesifikasi kompatibilitas, ada daftar platform yang didukung oleh blender dengan secara teratur diuji oleh tim pengembangan. Adapun daftar-daftar platform yang didukung bisa lebih lanjut dibuka pada [7].

Sebagai proyek berbasis umum di bawah GNU General Public License (GPL), Publik diberdayakan untuk membuat perubahan kecil maupun besar untuk basis kode, yang mengarah ke fitur baru, perbaikan bug, dan membuat tambahan yang lebih baik. [2]

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas mengenai analisis dan perancangan yang akan digunakan untuk menyelesaikan tugas akhir. Bab ini dibagi menjadi dua bagian. Pertama membahas mengenai penerapan algoritma alpha beta pruning sebagai kecerdasan buatan dalam permainan pawn battle. Bagian ini berisi data masukan, proses, serta keluaran yang dihasilkan, serta algoritma dan heuristic value yang digunakan.

Bagian kedua berisi perancangan permainan menggunakan kecerdasan buatan yang telah dibuat pada proses sebelumnya yang berisi analisis dan perancangan sistem.

3.1 Analisis Sistem

Permainan video berkembang sangat pesat akibat tuntutan perkembangan zaman. Masyarakat berminat terhadap permainan video yang tentunya menyenangkan. Apalagi dalam permainan pawn battle yang cenderung membutuhkan strategi dan pemikiran yang dalam, dibutuhkan lawan yang tepat sebagai teman berpikir dan berlatih kemampuan berpikir strategis. Sehingga dibutuhkan algoritma yang baik dalam menentukan tingkat kesulitan bermain yang membuat pemain merasa nyaman selama bermain .

Aplikasi ini dibangun dengan tujuan membantu para pemain dapat menikmati permainan yang variatif dan otomatis sekaligus dapat menambah ilmu pembelajaran yang edukatif. Tingkatan kesulitan dalam permainan dapat dipilih secara bebas oleh pemain sendiri sehingga pemain mampu berdaptasi sesuai dengan tingkat kesulitan masing-masing. Data yang digunakan untuk membedakan level adalah kedalaman node yang di expand dalam algoritma alpha beta pruning. Dengan kata lain, semakin tinggi level yang dipilih maka semakin dalam pula node yang di expand. Sehingga sang kecerdasan buatan mampu berpikir beberapa langkah lebih jauh sesuai kedalaman node. Sedangkan heuristic value yang diambil

mencakup 10 kondisi dalam permainan. 10 kondisi tersebut antara lain:

1. *Pawn en Prise*
2. Menghitung Pion
3. Pion yang Terlewat
4. Kondisi Akhir (Menang, Kalah atau *Stalemate*)
5. Mayoritas
6. Mayoritas yang lumpuh
7. Kombinasi *Backthrough*
8. Jarak Maksimal Pion
9. Pion yang Terisolasi
10. Pion yang Bertumpuk

Penulis menggunakan teknologi game engine Unity 5 dengan Bahasa pemrograman C# untuk membantu pengembangan permainan. Sedangkan asset yang ada dalam permainan dibuat dan dimodifikasi dengan menggunakan kakas blender.

3.2 Perancangan Permainan

3.2.1 Deskripsi Umum Perangkat Lunak

Tugas akhir yang akan dikembangkan adalah sebuah permainan 3D bergenre strategi dan turn based. Level permainan dipilih dengan bebas oleh pemain sesuai yang diinginkan. Level yang dipilih dengan bebas dimaksudkan agar pemain mampu menyesuaikan diri dan beradaptasi sesuai dengan kemampuan pemain saat ini. Sehingga diharapkan pemain mampu bermain dengan lawan yang setingkat dengan kemampuannya dan membantu pemikiran strategis. Permainan ini akan dijalankan dan dimainkan pada perangkat system operasi Windows berbasis desktop.

Pengguna utama dari permainan ini adalah semua orang yang ingin berlatih permainan catur dan ingin bermain battle pawn. Pemain dapat memilih tiga tingkatan level yakni **Easy**, **Medium** dan **Hard**. Pada level **Easy** pemain akan melawan NPC dengan tingkat pengetahuan 2 langkah didepan pemain. Pada level ini pemain yang sudah terbiasa dengan permainan catur akan mudah mengalahkan

NPC, karena permainan catur yang kompleks membuat pemain terbiasa memikirkan lebih dari 3 langkah kedepan. Namun untuk pemain yang tidak pernah bermain catur sebelumnya, level ini akan sangat sesuai sebagai latihan, karena di dengan pengetahuan 2 langkah ke depan, NPC hanya akan bermain cara-cara dasar bermain catur.

Selanjutnya pada level **Medium** NPC akan mengetahui 4 langkah di depan pemain. Di level ini, NPC mulai menerapkan dasar strategi bermain, karena dengan pengetahuan 4 langkah kedepan, NPC akan memikirkan semua langkah yang mungkin terjadi selama 4 langkah kedepan. Artinya, dengan lebih banyak kemungkinan langkah yang dihitung, akan membuat NPC lebih bisa memilih strategi terbaik yang diterapkan untuk melawan pemain. Namun pada pada level ini gerakan NPC akan sedikit lebih lama dari level easy.

Terakhir, pada level **Hard**, NPC akan menghitung 5 langkah di depan pemain. Di level ini, NPC akan memilih strategi terbaik yang dimiliki dalam kemungkinan langkah yang sangat bervariatif. Namun, dalam level ini, NPC akan bergerak jauh lebih lama dari sebelumnya, karena dengan menghitung semua kemungkinan di 5 langkah selanjutnya akan membuat NPC bekerja lebih keras daripada level-level sebelumnya.

Sebagai gambaran, rata-rata kemungkinan gerakan dalam 1 kali langkah adalah 16 langkah. Maka di level easy, NPC akan menghitung setidaknya 16^2 atau 256 kemungkinan langkah. Di level medium, NPC akan menghitung 16^4 atau 65.536 kemungkinan langkah. Di level hard, NPC akan menghitung 16^5 atau 1.048.576 kemungkinan langkah. Dengan biaya penghitungan langkah yang besar inilah, membuat NPC dengan tingkat kesulitan tinggi membutuhkan waktu yang relatif lama. Aturan bermain permainan ini sama dengan permainan catur tradisional, namun pemain hanya menggerakkan bidak pion yang hanya bisa melangkah kedepan dan tidak bisa mundur.

3.2.2 Spesifikasi Kebutuhan Fungsional

Berdasarkan deskripsi umum sistem, maka disimpulkan bahwa kebutuhan fungsional dari aplikasi ini hanya ada satu yaitu bermain game *battle pawn*.

3.2.3 Spesifikasi Kebutuhan Non-Fungsional

Terdapat beberapa kebutuhan non-fungsional yang apabila dipenuhi, dapat meningkatkan kualitas dari permainan ini. Berikut daftar kebutuhan non-fungsional:

1. Kebutuhan *Memory*

Permainan ini harus mampu dimainkan secara lancar, tidak ada lag dan nyaman di mata. Oleh karena itu, dengan perhitungan kemungkinan langkah yang bervariatif, dipengaruhi oleh spesifikasi Random Access Memory komputer yang akan memainkan game ini.

2. Kebutuhan Grafis

Kenyamanan bermain berbanding lurus dengan kualitas grafis yang disajikan dalam permainan. Efek seperti animasi merupakan salah satu daya tarik dalam suatu permainan. Efek-efek ini bisa membuat drop rate fps dan permainan melambat (lag), karena membutuhkan tambahan komputasi.

3.2.4 Karakteristik Pengguna

Berdasarkan deskripsi umum diatas, maka dapat diketahui bahwa pengguna yang akan menggunakan aplikasi ini ada dua orang, yaitu pemain yang memainkan permainan, dan pengembang. Karakteristik pengguna tercantum dalam Tabel 3.1.

Tabel 3.1 Karakteristik Pengguna

Nama Aktor	Tugas	Hak Akses Aplikasi	Kemampuan yang harus dimiliki
Pemain	Pihak luar yang memainkan permainan.	Memainkan permainan	Tidak ada

3.3 Perancangan Sistem

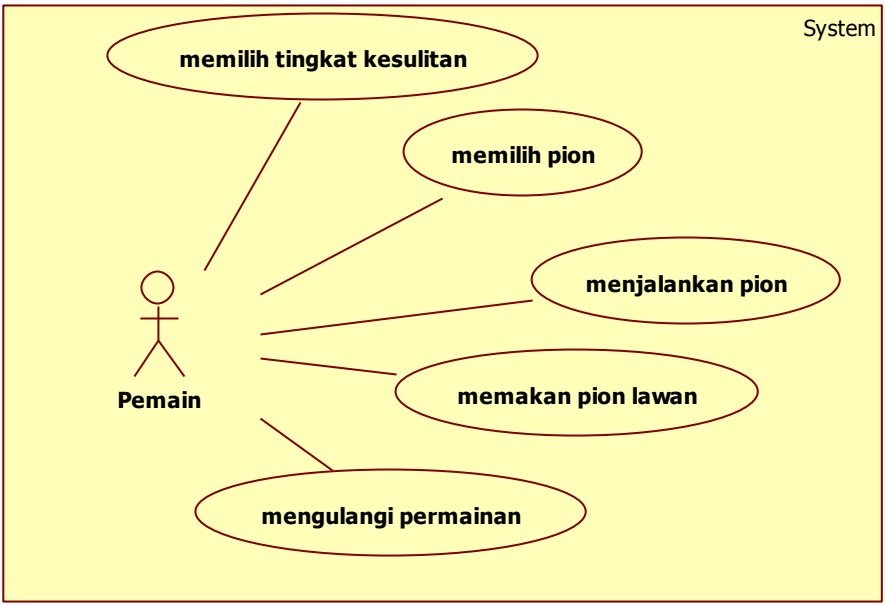
Tahap perancangan dalam subbab ini dibagi menjadi beberapa bagian yaitu perancangan diagram kasus penggunaan, perancangan skenario kasus penggunaan, perancangan antarmuka, perancangan kontrol permainan, perancangan alur permainan, perancangan *heuristic value* dan perancangan *Alpha beta pruning* sesuai dengan *rule-based strategy*.

3.3.1 Perancangan Diagram Kasus Penggunaan

Dalam aplikasi tugas akhir ini, terdapat lima kasus penggunaan. Lima kasus penggunaan ini diantaranya adalah memilih tingkat kesulitan, memilih pion, menjalankan pion, memakan pion lawan dan mengulangi permainan. Adapun kasus penggunaan NPC dijadikan satu dengan beberapa kasus penggunaan karena sebagian gerakan yang dilakukan NPC sama persis dengan yang mampu dilakukan pengguna. Pengguna atau entitas luar dari sistem adalah pemain yang memainkan permainan.

3.3.2 Perancangan Skenario Kasus Penggunaan

Kasus penggunaan yang terdapat didalam sistem dicantumkan pada Gambar 3.1.



Gambar 3.1 Diagram kasus aplikasi

Penjelasan dari masing-masing kasus penggunaan dicantumkan pada Tabel 3.2. Tabel tersebut berisi penjelasan skenario yang akan dilakukan ketika pengujian.

Tabel 3.2 Skenario Kasus Penggunaan

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
1	UC-001	Memilih tingkat kesulitan	Untuk menu pemain memilih tingkat lawan yang akan dihadapi
2	UC-002	Memilih pion	Untuk memillih pion yang dimiliki untuk selanjutnya dimainkan
3	UC-003	Menjalankan pion	Untuk menggerakkan pion yang telah dipilih sebelumnya kearah yang dituju pemain sesuai *rule*.
4	UC-004	Memakan pion lawan	Untuk menghilangkan pion lawan dan mengganti tempat tersebut dengan pion yang sebelumnya telah dipilih pemain
5	UC-005	Mengulangi permainan	Untuk memulai ulang permainan dengan tingkat kesulitan yang sama dengan yang sudah dipilih sebelumnya

3.3.2.1 Kasus Penggunaan Permainan

Penjelasan kasus penggunaan permainan untuk skenario UC-001 yakni Memilih tingkat kesulitan dijelaskan pada Tabel 3.3.

Tabel 3.3 Skenario Kasus Penggunaan Menjawab Soal Awal

Nama Kasus Penggunaan	Memilih tingkat kesulitan
Kode	UC-001
Deskripsi	Kasus penggunaan dimana aktor memilih tingkat kesulitan sebelum permainan dimulai.
Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan muncul layar **Menu**.

Alur Normal	1. Pemain memilih tombol tingkat kesulitan yang tersedia dan menekan klik. 2. Sistem menampilkan kondisi awal permainan dimana pion pemain dan pion NPC saling berjajar berhadapan.
Alur Alternatif	1. Pemain memilih tombol ‘EASY’ 2. Pemain memilih tombol ‘MEDIUM’ 3. Pemain memilih tombol ‘HARD’

Selanjutnya penjelasan kasus penggunaan permainan untuk skenario UC-002 yakni Memilih pion dijelaskan pada Tabel 3.4.

Tabel 3.4 Skenario Kasus Penggunaan Memilih Pion

Nama Kasus Penggunaan	Memilih pion
Kode	UC-002
Deskripsi	Kasus penggunaan dimana aktor memilih salah satu pion yang dimiliki pemain untuk kemudian dijalankan.
Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan kondisi pion sudah berjajar rapi saling berhadapan.
Alur Normal	1. Pemain memilih salah satu pion yang dimiliki. 2. Sistem menampilkan tanda bahwa pion telah aktif dan siap dijalankan.
Alur Alternatif	1. Pemain menekan pion lain. 2. Sistem akan menampilkan tanda bahwa pion lain tersebut aktif.

Kemudian penjelasan kasus penggunaan permainan untuk skenario UC-003 yakni Menjalankan pion dijelaskan pada Tabel 3.5.

Tabel 3.5 Skenario Kasus Menjalankan Pion

Nama Kasus Penggunaan	Menjalankan pion
Kode	UC-003
Deskripsi	Kasus penggunaan dimana aktor menjalankan pion ketika permainan berlangsung.
Aktor	Pemain.
Kondisi Awal	Pion yang dipilih pemain sudah aktif.
Alur Normal	1. Pemain mengarahkan pointer ke area yang diinginkan. 2. Sistem menampilkan gerakan pion melangkah ke tempat yang dituju.
Alur Alternatif	1. Pemain menekan ke area yang tidak diperbolehkan dalam *rule*. 2. Sistem menampilkan tanda bahwa gerakan tersebut tidak diperbolehkan

Selanjutnya penjelasan kasus penggunaan permainan untuk skenario UC-004 yakni Memakan pion lawan dijelaskan pada Tabel 3.6.

Tabel 3.6 Skenario Kasus Penggunaan Memakan Pion Lawan

Nama Kasus Penggunaan	Memakan pion lawan
Kode	UC-004
Deskripsi	Kasus penggunaan dimana aktor pion yang dimiliki untuk menempati area lawan dan sistem menghapus pion lawan.
Aktor	Pemain.

Kondisi Awal	Pemain sudah memilih pion yang akan dimainkan.
Alur Normal	1 Pemain memilih area yang ditempati lawan 2 Sistem menampilkan gerakan pion berjalan kearah lawan. 3 Sistem menampilkan gerakan pion memukul lawan 4 Sistem menghapus pion lawan 5 Pion pemain menempati area pion lawan.

Kemudian penjelasan kasus penggunaan permainan untuk skenario UC-005 yakni Mengulangi permainan dijelaskan pada Tabel 3.7.

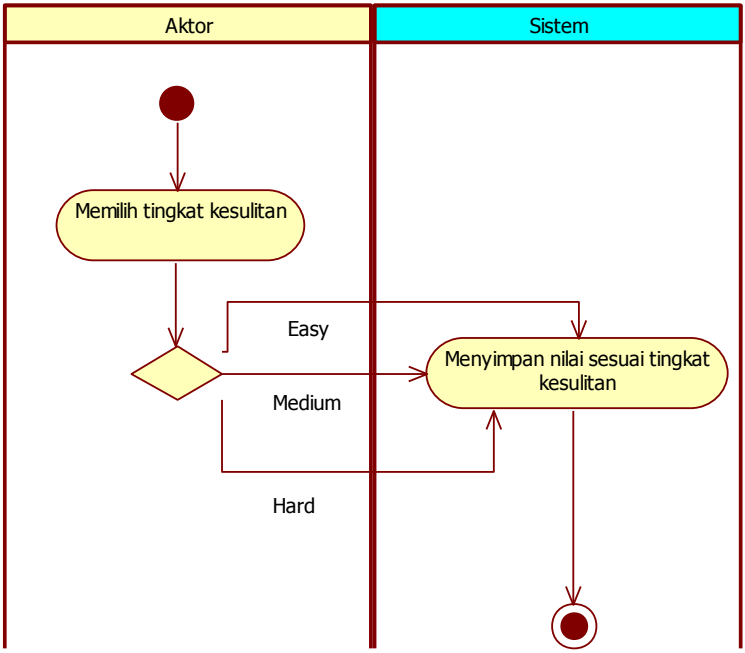
Tabel 3.7 Skenario Kasus Penggunaan Mengulangi Permainan

Nama Kasus Penggunaan	Mengulangi permainan
Kode	UC-005
Deskripsi	Kasus penggunaan dimana aktor memilih untuk mengulang permainan dari awal.
Aktor	Pemain.
Kondisi Awal	Pemain dalam kondisi kalah atau menang.
Alur Normal	1 Pemain memilih tombol ‘RESTART GAME’ dan menekan klik. 2 Sistem menampilkan kondisi awal permainan *Battle pawn*.

3.3.2.2 Diagram Aktivitas

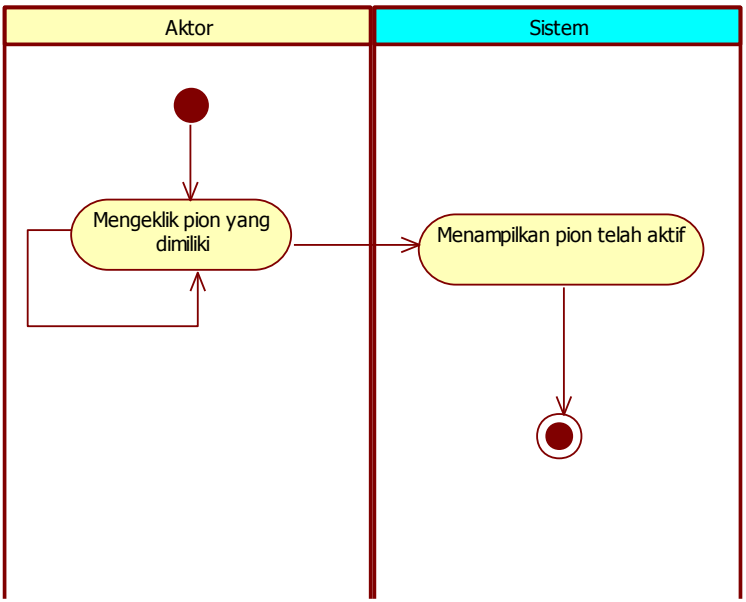
Diagram aktivitas menampilkan langkah-langkah normal yang harus dilakukan pemain untuk menjalankan studi kasus permainan dimulai dari awal permainan hingga kondisi akhir.

Diagram aktivitas untuk dari kasus penggunaan UC-001 yakni Memilih tingkat kesulitan dijelaskan pada Gambar 3.2.



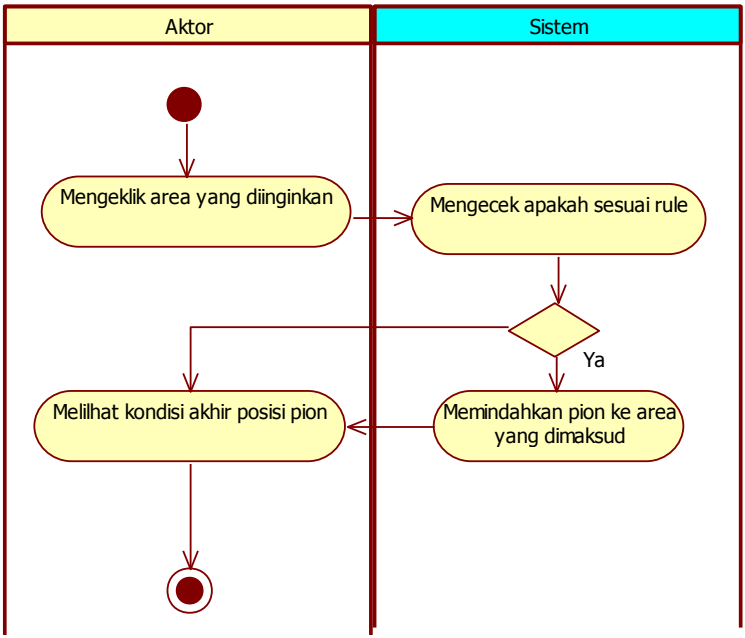
Gambar 3.2 Diagram aktivitas memilih tingkat kesulitan

Kemudian diagram aktivitas untuk dari kasus penggunaan UC-002 yakni Memilih pion dijelaskan pada Gambar 3.3.



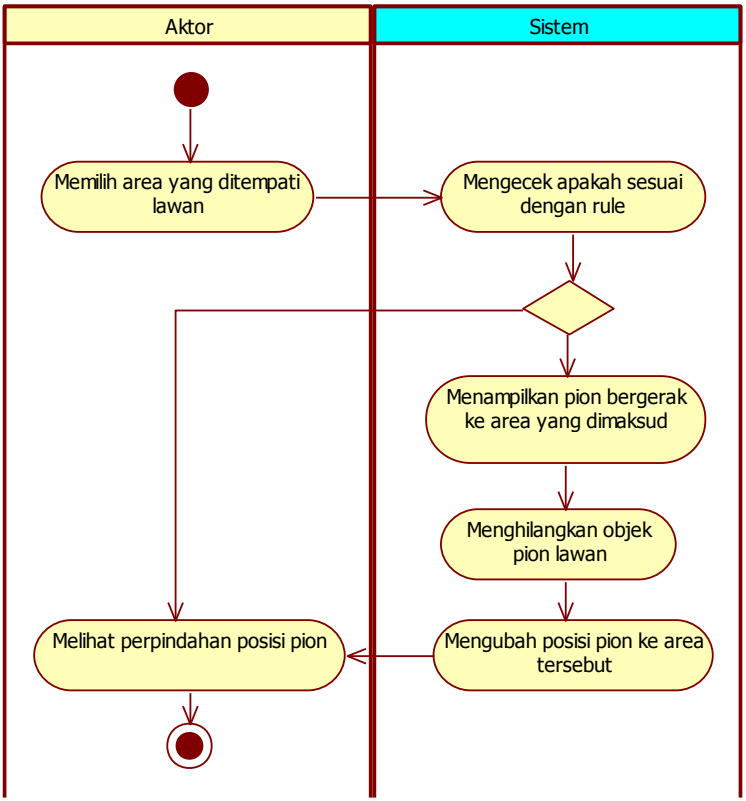
Gambar 3.3 Diagram aktivitas memilih pion

Kemudian diagram aktivitas untuk dari kasus penggunaan UC-003 yakni Menjalankan pion dijelaskan pada Gambar 3.4.



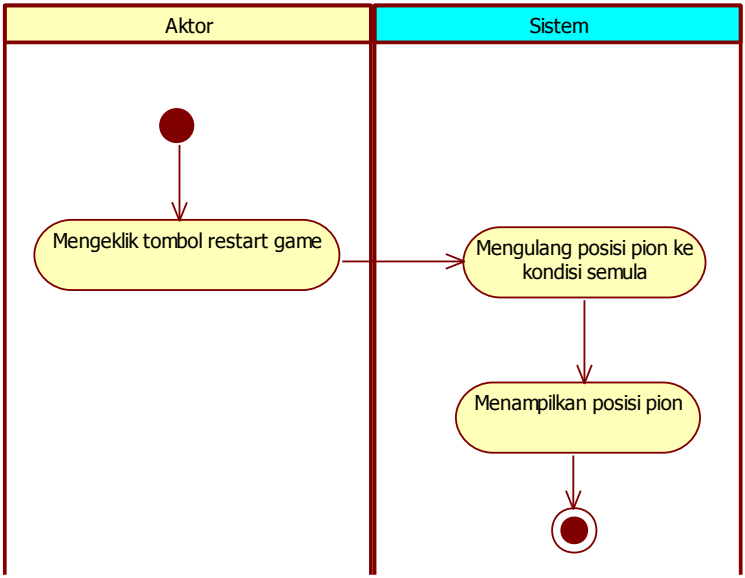
Gambar 3.4 Diagram aktivitas menjalankan pion

Kemudian diagram aktivitas untuk dari kasus penggunaan UC-004 yakni Memakan pion lawan dijelaskan pada Gambar 3.5.



Gambar 3.5 Diagram aktivitas memakan pion lawan

Selanjutnya diagram aktivitas untuk dari kasus penggunaan UC-005 yakni Mengulangi permainan dijelaskan pada Gambar 3.6.



Gambar 3.6 Diagram aktivitas mengulang permainan

3.3.3 Perancangan Antarmuka Pengguna

Subbab ini membahas bagaimana rancangan antarmuka pengguna yang akan digunakan untuk tugas akhir. Rancangan antarmuka yang dibahas meliputi ketentuan masukan dan rancangan jendela tampilan. Dalam aplikasi ini terdapat beberapa tampilan, yaitu **Main Menu**, **Kondisi Awal**, **Kondisi Menang**, **Kondisi Stalemate** dan **Kondisi Kalah**.

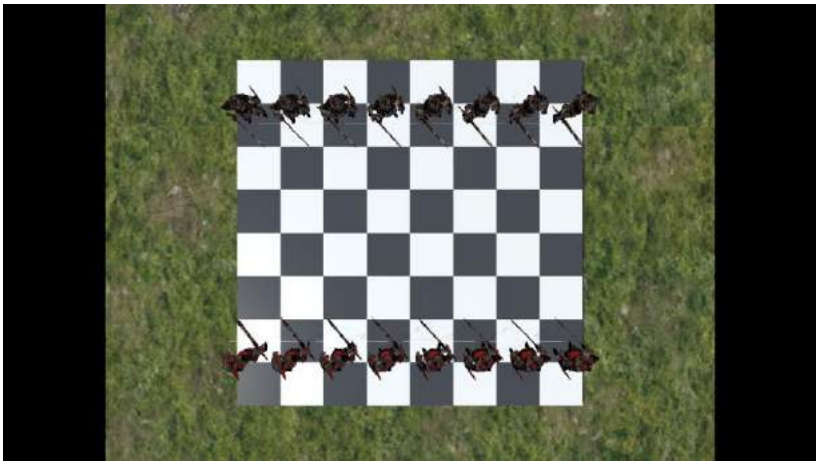
3.3.3.1 Tampilan Main Menu

Gambar 3.7 Tampilan Main Menu

Tampilan main menu merupakan tampilan yang pertama kali muncul ketika aplikasi dijalankan untuk yang pertama kalinya. Tampilan ini merupakan salah satu tampilan untuk interaksi pengguna yakni diminta untuk memilih tingkat kesulitan lawan NPC yang akan dihadapi.

3.3.3.2 Tampilan Kondisi Awal

Layar kondisi awal permainan ditampilkan ketika game berjalan untuk pertama kali. Tampilan ini merupakan tampilan awal permainan *battle pawn*. Dalam tampilan ini, pengguna melihat barisan pion yang dimiliki oleh pemain sedang berhadapan dengan lawan. Tampilan layar **Kondisi Awal Permainan** ditampilkan pada Gambar 3.12



Gambar 3.8 Tampilan Kondisi Awal Permainan

3.3.3.3 Tampilan Kondisi Menang

Tampilan kondisi menang merupakan tampilan yang ditampilkan ketika pion lawan sudah habis atau pion pemain sampai diseberang**.** Tampilan ini merupakan tampilan akhir. Pada tampilan ini pengguna dapat memilih tombol ‘RESTART GAME’ untuk masuk kembali memulai permainan dengan tingkat kesulitan yang sama. Atau tombol ‘MAIN MENU’ untuk bisa memilih tingkat kesulitan yang berbeda. Tampilan ini ditunjukkan pada Gambar 3.13.

Gambar 3.7 Tampilan Kondisi Menang

3.3.3.4 Tampilan Kondisi Stalemate

Tampilan kondisi stalemate merupakan tampilan yang ditampilkan ketika baik pion lawan maupun pion pemain tidak bisa bergerak sama sekali**.** Tampilan ini merupakan tampilan akhir. Pada tampilan ini pengguna dapat memilih tombol ‘RESTART GAME’ untuk masuk kembali memulai permainan dengan tingkat kesulitan yang sama. Atau tombol ‘MAIN MENU’ untuk bisa memilih tingkat kesulitan yang berbeda. Tampilan ini ditunjukkan pada Gambar 3.14.

Gambar 3.8 Tampilan Kondisi Stalemate

3.3.3.5 Tampilan Kondisi Kalah

Tampilan kondisi kalah merupakan tampilan yang ditampilkan ketika pion pemain sudah habis atau pion NPC sampai diseberang**.** Tampilan ini merupakan tampilan akhir. Pada tampilan ini pengguna dapat memilih tombol ‘RESTART GAME’ untuk masuk kembali memulai permainan dengan tingkat kesulitan yang sama. Atau tombol ‘MAIN MENU’ untuk bisa memilih tingkat kesulitan yang berbeda. Tampilan ini ditunjukkan pada Gambar 3.15

.

Gambar 3.9 Tampilan Kondisi Kalah

3.3.4 Perancangan Kontrol Permainan

Terdapat satu jenis kontrol dalam permainan yaitu menggunakan *mouse* dengan klik kiri ketika dijalankan pada desktop. Semua interaksi dari user hanya menggunakan satu kontrol tersebut.

3.3.5 Perancangan Alur Permainan

Alur permainan merupakan serangkaian proses yang harus diikuti pemain untuk memperoleh kemenangan. Peraturan permainan sama persis dengan permainan catur. Dimana pion yang dijalankan hanya bisa berjalan maju satu langkah dan memakan pion lawan yang berada di diagonal depan pion tersebut. Apabila pion berada dalam barisan awal, pion bisa dijalankan dua langkah kedepan. Ada kondisi tertentu yaitu *en passant*, dimana pion pemain melangkah dua langkah kedepan padahal disamping kirinya ada pion lawan. Kondisi menang ditentukan dengan habisnya semua pion lawan atau salah satu pion pemain berhasil sampai ke seberang. Kondisi terakhir

adalah *stalemate* dimana pemain lawan dan NPC tidak memiliki pion lagi untuk bisa dijalankan.

3.3.6 Perancangan *Heuristic Value*

Perancangan *heuristic value* merupakan komponen yang penting untuk diperhatikan karena *heuristic value* menentukan pintar atau tidaknya NPC. Dengan *heuristic value* yang salah, maka meskipun sedalam apapun *node* dibuka, tidak akan berpengaruh apa-apa. Bahkan akan membuat NPC semakin "bodoh". Berikut ini adalah 10 parameter yang dihitung untuk menentukan *heuristic value* sebuah *state*.

3.3.6.1 *Pawn en Prise*

Adalah kondisi apabila pion NPC berada pada baris ke-5 dan pion lawan yang berada dikiri dan kanan kolom ada pada baris awal, dan didepan pion tersebut tidak ada lawan yang menghadang.

Gambar 3.10 Ilustrasi *Pawn en Prise*

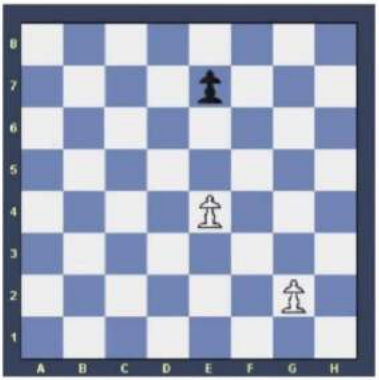
Apabila kondisi ini tercapai, maka pada implementasi algoritma, nilai node akan +3. Namun apabila dalam node, musuh mencapai kondisi seperti ini, maka nilai node akan -3.

3.3.6.2 Menghitung Pion

Menghitung banyak pion yang ada dalam *state* tersebut. Setiap pion yang dimiliki NPC, maka nilai node akan +5. Namun setiap pion yang dimiliki Player, nilai node akan -5.

3.3.6.3 Pion yang Terlewat

Adalah kondisi apabila disisi kanan dan kiri pion tidak ada pion lawan dan didepan pion tidak ada pion yang menghadang.



Gambar 3.11 Ilustrasi Pion yang Terlewat

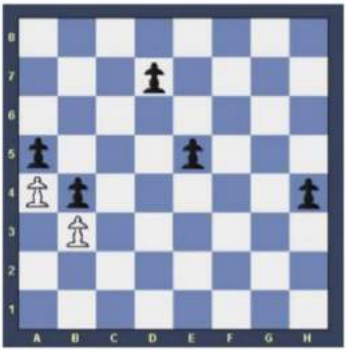
Apabila kondisi ini tercapai, maka nilai node akan dikalikan +5 dari jarak pion dengan baris paling belakang. Dan apabila Player berada dalam kondisi ini maka nilai node akan dikalikan -5 dari jarak pion Player ke baris paling depan.

3.3.6.4 Kondisi Akhir (Menang, Kalah atau Stalemate)

Kondisi menang adalah ketika salah satu pemain berhasil menghabiskan pion lawannya atau salah satu pionnya berhasil m encapai seberang.

Sebaliknya, pemain disebut kalah apabila pionnya habis atau salah satu pion lawan berhasil sampai kesebarang.

Sedangkan *stalemate* adalah kondisi dimana salah satu atau kedua pemain tidak dapat menggerakkan pionnya sesuai dengan aturan permainan. Kondisi ini adalah kondisi dimana permainan berakhir seri (tidak ada pemenang) atau dalam istilah catur, kita sering menyebut dengan istilah remis. Kondisi ini dapat terjadi walaupun salah satu pemain masih bisa memainkan pionnya sesuai dengan aturan permainan. Bahkan dapat pula terjadi meskipun salah satu pemain hanya memiliki 1 buah pion, dan pemain yang lain masih memiliki banyak sekali pion.



Gambar 3.12 Ilustrasi *Stalemate*

Apabila NPC menang, maka *heuristic value* suatu *state* akan diubah menjadi maximal. Apabila NPC kalah, *heuristic value* menjadi minimal. Dan apabila *stalemate*, nilai *heuristic value* menjadi 0;

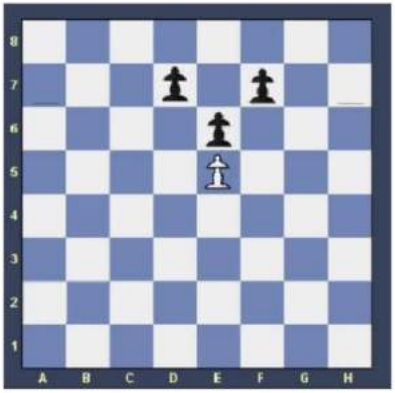
3.3.6.5 Mayoritas

Menghitung majoritas, adalah apabila ada pion NPC yang memiliki teman di sisi kiri dan kanan baris. Apabila tercapai kondisi ini, maka node akan dihitung dengan menjumlahkan jumlah pion

yang berwarna sama dikiri dan kanan baris, kemudian dikurangkan dengan jumlah musuh yang ada dikiri dan kanan baris.

3.3.6.6 Mayoritas yang Lumpuh

Adalah apabila ada pion yang berada di garis depan sehingga membuat pergerakan pion musuh dikanan dan kiri barisnya tidak menguntungkan.



Gambar 3.13 Ilustrasi Mayoritas yang Lumpuh

Apabila kondisi ini tercapai, maka jarak pion dengan garis akhir akan dikalikan +2. Dan apabila pion musuh mencapai kondisi ini, maka jarak pion lawan dengan garis akhir akan dikalikan -2.

3.3.6.7 Kombinasi Backthrough

Adalah kombinasi 3 pion sejajar yang berhadapan. Dimana pemain yang pionnya berada di garis depan, akan diuntungkan. Yakni menang, atau stalemate.

Gambar 3.20 Ilustrasi Kombinasi *Backthrough*

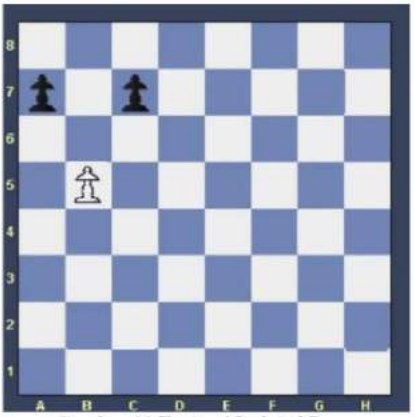
Apabila kondisi ini tercapai, maka nilai node akan ditambahkan dengan jarak pion NPC dengan garis awal dikurangi jarak pion Player dengan garis awal kemudian hasil akhir dikalikan +2.

3.3.6.8 Jarak Maksimal Pion

Menghitung jarak pion terjauh dari NPC dan mengurangi dengan jarak pion terjauh milik pemain.

3.3.6.9 Pion yang Terisolasi

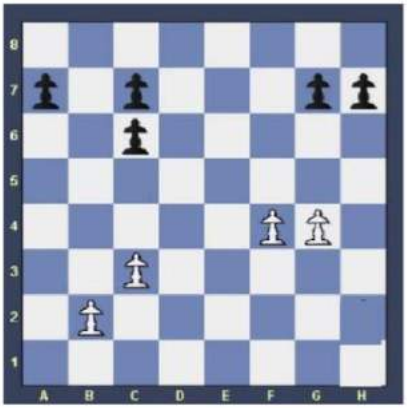
Adalah kondisi dimana pion “sendirian” sedangkan lawan sudah berada di garis depan. Apabila terjadi kondisi ini, akan menguntungkan pemain yang bidaknya berada di garis depan. Meskipun lawan memiliki pion yang lebih banyak, namun apabila “sendirian”, tetap menguntungkan pemain. Apabila terjadi kondisi seperti ini, maka nilai node adalah jarak antara pion dengan garis awal kemudian dikalikan +2.



Gambar 3.21 Ilustrasi Pion yang Terisolasi

3.3.6.10 Pion yang Bertumpuk

Kondisi dimana 2 atau lebih pion yang sama bertumpukan. Sehingga jumlah yang banyak tidak memberi pengaruh yang besar. Pergerakan pemain pun kurang maksimal.

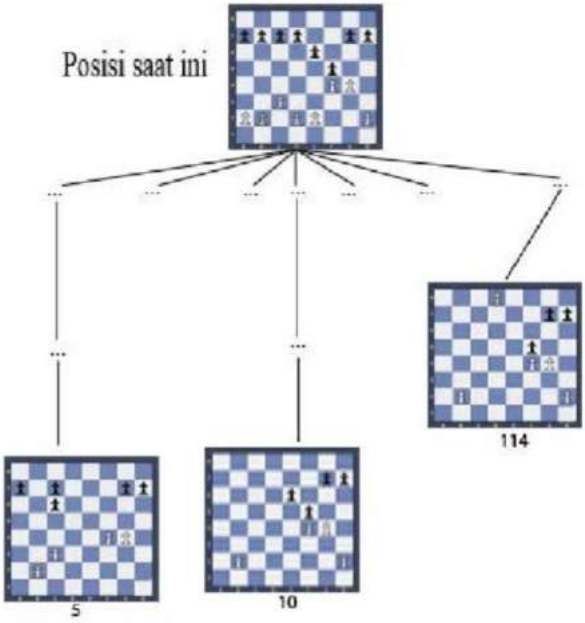


Gambar 3.22 Ilustrasi Pion yang bertumpuk

Apabila terjadi, maka setiap penumpukan pion akan -1.

3.3.7 Perancangan Algoritma Alpha Beta Pruning

Algoritma alpha beta pruning adalah salah satu algorima yang menerapkan pemodelan *tree* dan *depth first search* hingga kedalaman tertentu, kemudian membandingkan nilai dengan *node-node* lain dalam satu *level depth*. Apabila giliran NPC, maka yang dicari adalah nilai maksimal *node*. Sedangkan apabila giliran pemain, maka yang dicari adalah nilai minimal *node*. *Expand* suatu *node* dilakukan dengan memperhatikan aturan-aturan yang sudah ditentukan *rule-based*.



Gambar 3.23 Ilustrasi Penerapan *Alpha Beta Pruning* sesuai *rule-based strategy*

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan perangkat lunak. Di dalamnya mencakup proses penerapan dan pengimplementasian algoritma, dan antar muka yang mengacu pada rancangan yang telah dibahas sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi dari tugas akhir dijelaskan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat Keras	Prosesor: Pentium(R) Dual-Core CPU T4400 @ 2.20GHz Memori: 4 GB
Perangkat Lunak	Sistem Operasi: Microsoft Windows 8 64-bit Perangkat Pengembang: Unity 5

4.2 Implementasi Permainan

Implementasi dari masing-masing fungsi utama dituliskan menggunakan *pseudocode* berdasarkan antarmuka utama yang ada pada permainan. Penjelasan implementasi hanya berupa antar muka yang berhubungan dengan *rule-based* permainan *pawn* battle, penerapan algoritma alpha beta pruning dan pemilihan *heuristic value* diantaranya pemilihan Tingkat Kesulitan, pengecekan Langkah Sesuai Aturan, penerapan Algoritma *Alpha beta pruning* dan pemilihan *Heuristic Value*.

4.2.1 Implementasi Pemilihan Tingkat Kesulitan

Implementasi pembuatan layar **Tingkat Kesulitan** ditiliskan pada Gambar 4.1. Pada layar ini terdapat tiga buah tombol yaitu tombol ‘EASY’, ‘MEDIUM’ dan ‘HARD’. Tiga tombol tersebut menyimpan nilai tersendiri yang nantinya akan

digunakan sebagai maksimal *depth level* pada penerapan algoritma.

Implementasi pembuatan layar untuk memilih **Tingkat Kesulitan** tercantum di Gambar 4.1. Pada implementasi ini dapat dilihat bahwa setiap tombol memiliki nilai tersendiri.

```
// fungsi MenuController() //
Easy()
Medium()
Hard()
```

Gambar 4.1 *Pseudocode* pemilihan Tingkat Kesulitan

Fungsi ‘Easy()’ pada Gambar 4.2 merupakan fungsi yang digunakan untuk melakukan *set* nilai level menjadi 2.

```
// Fungsi Easy() //
level = 2

DepthLevel <- level
```

Gambar 4.2 *Pseudocode* fungsi untuk tombol Easy

Fungsi ‘Medium()’ pada Gambar 4.3 merupakan fungsi yang digunakan untuk melakukan *set* nilai level menjadi 4.

```
// Fungsi Medium() //
level = 4

DepthLevel <- level
```

Gambar 4.3 *Pseudocode* fungsi untuk tombol Medium

Fungsi ‘Hard()’ pada Gambar 4.2 merupakan fungsi yang digunakan untuk melakukan *set* nilai level menjadi 5.

```
// Fungsi Hard() //
level = 5

DepthLevel <- level
```

Gambar 4.4 *Pseudocode* fungsi untuk tombol Hard

4.2.2 Implementasi Pengecekan Langkah

Ketika permainan dimulai terdapat barisan pion pemain dan NPC yang saling berhadapan. Barisan pion pemain apabila ditekan klik maka akan mengaktifkan pion tersebut sehingga pion tersebut siap dijalankan. Ketika ada pion yang aktif, maka terdapat fungsi yang bisa menentukan semua gerakan pion yang mungkin.

Implementasi dimulai ketika pengguna telah memilih salah satu pion. Pertama program memeriksa apakah pion yang diklik benar-benar pion milik pemain atau milik NPC. Kemudian pion akan aktif dan system akan menghitung semua kemungkinan gerakan pion yang mungkin. Selanjutnya setelah semua gerakan disimpan, system akan menampilkan tanda bahwa pion tersebut telah aktif dan siap untuk berjalan. Sistem akan menunggu sampai pemain mengeklik suatu area. Setelah pemain memilih suatu area, maka system akan membandingkan, apakah area tersebut merupakan bagian dari *list* gerakan yang mungkin. Jika benar, maka system akan memindahkan posisi pion yang aktif tersebut ke area yang ditentukan pemain. *Pseudocode* dari implementasi dijelaskan pada Gambar 4.5.

```
// fungsi OnClick() //
If(pion.turnplayer = this.turn)
   SetActive(pion)
```

Gambar 4.3 *Pseudocode* pembentukan soal *training*

Fungsi ‘AvailableMoves()’ pada Gambar 4.6 merupakan fungsi yang digunakan untuk menghitung semua langkah yang mungkin dilakukan oleh pion.

```
// Fungsi AvailableMoves(pion) //
ListMoves <- null
posisi <- pion.position
If(CheckFront(posisi) = 1)
   ListMoves <- Front(posisi)
If(posisi = posisi_awal and Check2Front(posisi) = 1 )
```

```
   ListMoves <- Front(Front(posisi))
If(CekLawanKiri(posisi) = 1)
   ListMoves <- DiagonalKiri(posisi)
If(CekLawanKanan(posisi) = 1)
   ListMoves <- DiagonalKanan(posisi)
If(CekEnpassant(posisi) = 1)
   ListMoves <- DiagonalPassant(posisi)
```

Gambar 4.4 *Pseudocode* kemungkinan langkah pion

Fungsi 'Bergerak()' pada Gambar 4.7 meupakan fungsi memindahkan pion yang aktif ke tempat yang diinginkan. Fungsi ini cukup membandingkan apakah posisi yang diinginkan terdapat dalam ListMoves atau tidak. Jika ternyata dalam ListMoves ada posisi yang diinginkan, maka pion akan dipindahkan ke tempat tersebut.

```
// fungsi Bergerak(pion, posisiTujuan) //
if(pion.IsActive = 1)
   for(i in range(ListMoves))
      if(posisiTujuan = ListMoves[i])
          if(CekLawan(posisiTujuan) = 1)
             MakanPion(posisiTujuan)
          pion.posisi <- posisiTujuan
          break
```

Gambar 4.5 *Pseudocode* pion bergerak

Fungsi 'MakanPion()' pada Gambar 4.8 merupakan fungsi untuk menghilangkan pion dari *list of pion* sehingga untuk selanjutnya pion tersebut tidak diperhitungkan lagi.

```
// fungsi MakanPion(posisiTarget) //
Foreach(pioniter in ListOfPion)
   If(pioniter.posisi = posisiTarget)
      ListOfPion.Remove(pioniter)
      Destroy(pioniter)
```

Gambar 4.6 *Pseudocode* memakan pion

4.2.3 Implementasi Algoritma *Alpha Beta Pruning*

Algoritma *Alpha beta pruning* adalah algoritma pemodelan *tree* yang menggunakan *depth first search* sebagai metode pencarian *node* terbaiknya. Dalam kasus ini, yang dimaksud dengan *node* adalah satu set pion pemain dan NPC beserta semua 64 area papan hitam putih sebagai alas permainan. *Node* ini diambil dari semua langkah yang mungkin dilakukan oleh pemain dan NPC. Sehingga dihasilkan nilai optimal tergantung seberapa dalam *node* akan di*expand*. Sebenarnya, algoritma ini sama dengan algoritma *MinMax*, namun yang membuat algoritma ini lebih baik ialah karena algoritma ini menggunakan nilai *alpha* dan *beta* yang didapat dari nilai tertinggi dan terendah suatu node yang sebelumnya pernah dihitung. Apabila terjadi nilai *alpha* lebih tinggi dari nilai *beta*,. Maka kegiatan *expanding node* akan di potong dan tidak dilanjutkan. Karena kita tidak perlu berharap dan menghitung kemungkinan dimana pengguna permainan melakukan kesalahan dalam bermain. Pseudocode implementasi ini dijelaskan pada Gambar 4.9.

```
// fungsi Alphabeta(root, alpha, beta, depth, player)
if depth = 0 or node is a terminal node
   return HitungState(node)
      if player = 'black'
          v := -∞
          foreach(pioniter in node.pion)
             foreach AvailableMoves(pioniter)
                child <- (Bergerak(pioniter))
                v := max(v, alphabeta(child, depth - 1,
α, β, 'white'))
              α := max(α, v)
              if β ≤ α
                  break (* prune *)
          return v
      else
          v := ∞
```

```
          foreach(pioniter in node.pion)
             foreach AvailableMoves(pioniter)
                child <- (Bergerak(pioniter))
                v := min(v, alphabeta(child, depth - 1,
α, β, 'black'))
              β := min(β, v)
              if β ≤ α
                  break (* prune *)
          return v
```

Gambar 4.9 *Pseudocode* penerapan Alpha Beta Pruning

4.2.4 Implementasi Pemilihan *Heuristic Value*

Pada pemain telah melakukan giliran, maka giliran NPC yang bermain. NPC akan melakukan penghitungan setiap *state* yang didapat dari fungsi AvailableMoves() sebelumnya. *Heuristic value* merupakan komponen penting dalam membuat kecerdasan buatan dalam permainan ini, karena apabila nilai *heuristic value* diambil dengan tidak benar, gerakan pion yang dilakukan oleh NPC tidak akan optimal.

Implementasi dimulai ketika pengguna telah selesai melakukan giliran dengan menjalankan salah satu pion yang dipunyai. Pertama program akan menjalankan algoritma Alpha beta pruning hingga kedalaman tertentu, kemudian ketika *node* telah mencapai kedalaman yang diinginkan, fungsi HitungState() baru dijalankan. Fungsi ini menghitung nilai yang didapat dari set pion lawan dan NPC yang dimiliki di tiap-tiap *state*. Ada 10 hal yang memengaruhi pengambilan nilai dalam fungsi ini. Antara lain:

1. *Pawn en Prise*
2. Menghitung Pion
3. Pion yang Terlewat
4. Kondisi Akhir (Menang, Kalah atau *Stalemate*)
5. Mayoritas
6. Mayoritas yang lumpuh
7. Kombinasi *Backthrough*

8. Jarak Maksimal Pion
9. Pion yang Terisolasi
10. Pion yang Bertumpuk

Seperti yang sudah dijelaskan sebelumnya. *Pseudocode* untuk mencari *heuristic value* tercantum pada Gambar 4.10.

```
// fungsi HitungState(node) //
// menghitung heuristic value suatu state
result <- 0
result += jumlah_pion(node.ListOfPion);
result += jarakmaxplayer(node.ListOfPion);
result += kondisiakhir(node.ListOfPion);
result += pawnenterprise(node.ListOfPion);
result += majorities(node.ListOfPion);
result += passedpawn(node.ListOfPion);
result += disabledmajority(node.ListOfPion);
result += breakthroughC(node.ListOfPion);
result += doubledpawn(node.ListOfPion);
result += isolatedPawn(node.ListOfPion);
return result
```

Gambar 4.10 *Pseudocode* mencari *heuristic value* suatu *state*

Fungsi jumlah_pion() adalah fungsi untuk menghitung selesih antar pion NPC dengan pengguna.atau dengan kata lain Menghitung Pion. *Pseudocode* fungsi ini dijelaskan pada gambar 4.11.

```
// fungsi jumlah_pion(ListOfPion)/
Result <- 0
foreach(Pion a in ListOfPion)
{
  if (a.player == "white")
      result <- result-5;
  else if (a.player == "black")
      result <- result+5;
}
return result;
```

Gambar 4.7 *Pseudocode* Menghitung Pion

Fungsi jarakmaxplayer() adalah fungsi untuk menghitung selesih jarak terjauh pion NPC dengan pengguna.atau dengan kata lain Jarak Maksimal Pion. *Pseudocode* fungsi ini dijelaskan pada gambar 4.12.

```
// fungsi jarakmaxplayer (ListOfPion)/
int result = 0;
float maxWhite = 8f;
float maxBlack = 8f;
float finishPutih = -5f;
float finishHitam = 2f;
foreach (Peon a in pion)
{
   if(a.player=="white")
   {
      if (maxWhite > (a.positionx - finishPutih))
         maxWhite = a.positionx - finishPutih;
   }
   if (a.player == "black")
   {
      if (maxBlack > (finishHitam - a.positionx))
         maxBlack = finishHitam - a.positionx;
   }
}
result = (int)(maxWhite - maxBlack);
return result;
```

Gambar 4.8 *Pseudocode* Jarak Maksimal Pion

Fungsi kondisiakhir() adalah fungsi untuk menghitung apakah *state* tersebut merupakan *state* menang kalah atau *stalemate*. *Pseudocode* fungsi ini dijelaskan pada gambar 4.13.

```
// fungsi kondisiakhir(ListOfPion)/
int result = 0;
foreach (Peon a in pion)
{
   if (a.player == "black")
      if (a.positionx == 2)
      {
         result = 60000;
         break;
      }
```

```
   if (a.player == "white")
      if (a.positionx == -5)
      {
         result = -60000;
         break;
      }}
return result;
```

Gambar 4.9 *Pseudocode* kondisi akhir

Fungsi pawnenterprise() adalah fungsi untuk menghitung kondisi *Pawn en prise* seperti yang dijelaskan sebelumnya. *Pseudocode* fungsi ini dijelaskan pada gambar 4.14.

```
// fungsi pawnenterprise(ListOfPion)/
int result = 0;
foreach (Peon a in pion)
{
   if (a.player == "white" && a.positionx == -2)
   {
      if (isExist(pion, a.positionx - 1, a.positionz)
== 0 && isExist(pion, a.positionx - 2, a.positionz)
== 0)
      {
         if(a.positionz < 5)
            if (isExist(pion, a.positionx - 2,
a.positionz + 1) == 1 && isExist(pion, a.positionx -
1, a.positionz + 1) == 0)
            result = result - 3;
         if(a.positionz > -2)
         if (isExist(pion, a.positionx - 2,
a.positionz - 1) == 1 && isExist(pion, a.positionx -
1, a.positionz - 1) == 0)
            result = result - 3;
      }
   }
   if (a.player == "black" && a.positionx == -1)
   {
      if (isExist(pion, a.positionx + 1, a.positionz)
== 0 && isExist(pion, a.positionx + 2, a.positionz)
== 0)
      {
         if (a.positionz < 5)
         if (isExist(pion, a.positionx + 2,
```

```
a.positionz + 1) == 1 && isExist(pion, a.positionx +
1, a.positionz + 1) == 0)
            result = result + 3;
         if (a.positionz > -2)
         if (isExist(pion, a.positionx + 2,
a.positionz - 1) == 1 && isExist(pion, a.positionx +
1, a.positionz - 1) == 0)
            result = result + 3;
       }
   }
}
        return result;
```

Gambar 4.10 ***Pseudocode Pawn en Prise***

Fungsi majorities() adalah fungsi untuk menghitung banyaknya mayoritas dari pion NPC dan pemain dalam *state* tersebut. *Pseudocode* fungsi ini dijelaskan pada gambar 4.15.

```
// fungsi majorities(ListOfPion)/
int result = 0;
int whiteVal = 0;
int blackVal = 0;
foreach(Peon a in pion)
{
    if (a.player == "white")
        whiteVal += isExistinline(pion, a.player,
a.positionz);
    if (a.player == "black")
        blackVal += isExistinline(pion, a.player,
a.positionz);
}
result = blackVal - whiteVal;
return result;
```

Gambar 4.11 ***Pseudocode Majorities***

Fungsi passedpawn() adalah fungsi untuk menghitung banyaknya pion yang terlewat dari pion NPC dan pemain dalam *state* tersebut. *Pseudocode* fungsi ini dijelaskan pada gambar 4.16.

```
// fungsi passedpawn(ListOfPion)/
int result = 0;
```

```
float maxWhite = 8f;
float maxBlack = 8f;
float finishPutih = -5f;
float finishHitam = 2f;
foreach (Peon a in pion)
{
    if (isTouchdown(pion, a) == 1)
    {
        if (a.player == "white")
        {
            if (maxWhite > (a.positionx -
finishPutih))
                maxWhite = a.positionx - finishPutih;
        }
        if (a.player == "black")
        {
            if (maxBlack > (finishHitam -
a.positionx))
                maxBlack = finishHitam - a.positionx;
        }
    }
}
result = (int)((maxWhite - maxBlack)*5);
return result;
```

Gambar 4.12 ***Pseudocode Majorities***

Fungsi disabledmajorities() adalah fungsi untuk menghitung banyaknya *disabledmajorities* dari pion NPC dan pemain dalam *state* tersebut. *Pseudocode* fungsi ini dijelaskan pada gambar 4.18.

```
// fungsi disabledmajorities(ListOfPion)/
int result = 0;
float maxWhite = 8f;
float maxBlack = 8f;
float finishPutih = -5f;
float finishHitam = 2f;
foreach (Peon a in pion)
{
    if(isDepanlawan(pion,a)==1)
    {
        if(sampingAman(pion,a)==1)
        {
```

```
            if (a.player == "white")
            {
                if (maxWhite > (a.positionx -
finishPutih))
                    maxWhite = a.positionx -
finishPutih;
            }
            if (a.player == "black")
            {
                if (maxBlack > (finishHitam -
a.positionx))
                    maxBlack = finishHitam -
a.positionx;
            }
        }
    }
}
result = (int)((maxWhite - maxBlack)*2);
return result;
```

Gambar 4.13 ***Pseudocode Disabled Majorities***

Fungsi breakthroghC() adalah fungsi untuk menghitung banyaknya *breakthrough combination* antara pion NPC dengan pion pemain dalam *state* tersebut. *Pseudocode* fungsi ini dijelaskan pada gambar 4.18.

```
// fungsi breakthroughC(ListOfPion)/
int result = 0;
float maxWhite = 8f;
float maxBlack = 8f;
float finishPutih = -5f;
float finishHitam = 2f;
foreach (Peon i in pion)
{
    if(isLineDepanlawan(pion, i) == 1)
    {
        if(isSampingTeman(pion, i) == 1)
        {
            Peon lawan = null;
            lawan = getPosDepan(pion, i);
            if(isSampingTeman(pion, lawan) == 1)
            {
                if (i.player == "white")
```

```
                {
                    if (maxWhite > (i.positionx -
finishPutih))
                        maxWhite = i.positionx -
finishPutih;
                }
                if (i.player == "black")
                {
                    if (maxBlack > (finishHitam -
i.positionx))
                        maxBlack = finishHitam -
i.positionx;
                }
            }
        }
    }
}
result = (int)((maxWhite - maxBlack)*2);
return result;
```

Gambar 4.14 ***Pseudocode Breakthrough Combination***

Fungsi doubedpawn() adalah fungsi untuk menghitung banyaknya pion yang bertumpukan dalam 1 baris antara pion NPC dengan pemain dalam *state* tersebut. *Pseudocode* fungsi ini dijelaskan pada gambar 4.19.

```
// fungsi doubledpawn(ListOfPion)/
int result = 0;
foreach (Peon i in pion)
{
    if(isDepanteman(pion, i) == 1)
    {
        if(i.player == "black")
            result = result-1;
        if(i.player == "white")
            result = result+1;
    }
}
return result;
```

Gambar 4.15 ***Pseudocode Doubled Pawn***

Fungsi isolatedpawn() adalah fungsi untuk menghitung banyaknya pion yang tidak bisa berbuat banyak karena dalam posisi terhimpit antara pion NPC dengan pemain dalam *state* tersebut. *Pseudocode* fungsi ini dijelaskan pada gambar 4.20.

```
// fungsi isolatedpawn(ListOfPion)/
int result = 0;
int flag = 0;
float maxWhite = 8f;
float maxBlack = 8f;
float finishPutih = -5f;
float finishHitam = 2f;
foreach (Peon i in pion)
{
    if(isLineDepanlawan(pion, i) == 0)
    {
        if(i.positionz > -2)
        if(checkLineMusuh(pion, i.player,
i.positionx, i.positionz-1) == 1)
        {
            if(i.positionz > -1)
                if(checkLineMusuh(pion, i.player,
i.positionx, i.positionz-2) == 0)
                    flag = 1;
            else flag = 1;
        }
        if(i.positionz < 5)
        if(checkLineMusuh(pion, i.player,
i.positionx, i.positionz+1) == 1)
        {
            if(i.positionz < 4)
                if(checkLineMusuh(pion, i.player,
i.positionx, i.positionz+2) == 0)
                    flag = 1;
            else flag = 1;
        }
    }
    if(flag == 1)
    {
        if (i.player == "white")
        {
            if (maxWhite > (i.positionx -
finishPutih))
```

```
                maxWhite = i.positionx - finishPutih;
            }
            if (i.player == "black")
            {
                if (maxBlack > (finishHitam -
i.positionx))
                maxBlack = finishHitam - i.positionx;
            }
        }
}
result = (int)((maxWhite - maxBlack)*2);
return result;
```

Gambar 4.16 ***Pseudocode Isolated Pawn***

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Proses pengujian dilakukan dengan survei langsung kepada pengguna.

5.1 Lingkungan Uji Coba

Lingkungan pelaksanaan uji coba meliputi perangkat keras dan perangkat lunak yang akan digunakan pada sistem ini. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam rangka uji coba perangkat lunak ini dicantumkan pada Tabel 5.1.

Tabel 5.1 Lingkungan Ujicoba Perangkat Lunak

Perangkat Keras	Prosesor: Pentium(R) Dual-Core CPU T4400 @ 2.20GHz Memori: 4 GB
Perangkat Lunak	Sistem Operasi: Microsoft Windows 8 64-bit Perangkat Pengembang: Adobe Profesional Flash CC

5.2 Pengujian Penerapan *Rule-based strategy*

Pengujian ini dilakukan untuk menguji apakah *rule-based strategy* sudah benar-benar diimplementasikan dan bekerja dengan semestinya. Selain itu juga untuk mengetahui kesesuaian keluaran dari setiap langkah dari setiap terhadap skenario yang dipersiapkan.

5.2.1 Skenario Pengujian Langkah Awal

Skenario Pengujian hasil implementasi *rule-based strategy* dengan memprediksi langkah awal tiap level ini digunakan untuk memberikan tahap-tahap dalam pengujian sistem. Skenario ini tertera pada Tabel 5.2.

Tabel 5.2 Pengujian Permainan Berdasarkan Langkah Awal

	UFG01
Kondisi Awal	Masing-masing pion sudah menempati barisan masing-masing
Prosedur Pengujian	Pengguna memainkan permainan hingga 3 langkah pertama dari masing-masing level.
Hasil yang diharapkan	Sistem telah berjalan sebagaimana mestinya dan penerapan *rule-based strategy* telah diimplementasikan dengan benar.
Hasil yang diperoleh	Pion yang dijalankan NPC menempati area yang sebelumnya telah diperhitungkan.
Kesimpulan.	Pengujian berhasil.

5.2.1.1 P01: Pengujian Level Easy

Pengujian dimulai ketika pengguna sudah memasuki layar **Main Menu**.

Gambar 5.1 Tampilan Main Menu

Ketika pemain melangkahkan pionnya, dalam algoritma yang diterapkan, jika level easy dimainkan hanya akan melakukan *expand node* dengan *depth level* 2. Sehingga karena hanya melakukan *expand node* sebanyak 2 langkah, semua gerakan yang mungkin menghasilkan nilai yang sama. Dalam algoritma ini, jika ada nilai *node* yang sama, maka yang diambil adalah node yang peratama kali dihitung. Sehingga hasil *node* dalam level easy untuk 3 langkah pertama pasti pion hitam paling bawah dijalankan 2 kali. Kemudian pion akan mengikuti langkah pemain untuk menghalangi pion pemain maju lebih jauh, selama pion tersebut tidak dalam keadaan terancam.

Gambar 5.2 Langkah pertama pada level easy

Gambar 5.3 Langkah kedua pada level easy

Gambar 5.4 Langkah ketiga pada level easy

5.2.1.2 P02: Pengujian Level Medium

Pengujian dimulai ketika pengguna sudah memasuki layar **Main Menu** kemudian pemain mengeklik tombol *medium*.

Gambar 5.5 Tampilan Main Menu

Ketika pemain melangkahkan pionnya, dalam algoritma yang diterapkan, jika level medium dimainkan hanya akan melakukan *expand node* dengan *depth level* 4. Sehingga dengan melakukan *expand node* sebanyak 4 langkah, semua gerakan yang mungkin dikalkulasi oleh NPC dan menghasilkan gerakan yang bermacam-macam. Namun, dapat diperkirakan bahwa gerakan pion akan lebih banyak 1 langkah kedepan dibandingkan 2 langkah.

Gambar 5.6 Langkah pertama pada level medium

Gambar 5.6 Langkah kedua pada level medium

Gambar 5.7 Langkah ketiga pada level medium

5.2.1.3 P03: Pengujian Level Hard

Pengujian dimulai ketika pengguna sudah memasuki layar **Main Menu** kemudian pemain mengeklik tombol *hard.*

Gambar 5.8 Tampilan Main Menu

Ketika pemain melangkahkan pionnya, dalam algoritma yang diterapkan, jika level *hard* dimainkan hanya akan melakukan *expand node* dengan *depth level* 5. Sehingga dengan melakukan *expand node* sebanyak 5 langkah, semua gerakan yang mungkin dikalkulasi oleh NPC dan menghasilkan gerakan yang bermacam-macam. Namun, dapat diperkirakan bahwa gerakan pion akan lebih banyak menghindari pion termakan dan berusaha menjaga pion yang lain tetap aman.

Gambar 5.9 Langkah pertama pada level hard

Gambar 5.10 Langkah kedua pada level hard

Gambar 5.11 Langkah ketiga pada level hard

5.2.2 Skenario Pengujian Antar Kecerdasan Buatan

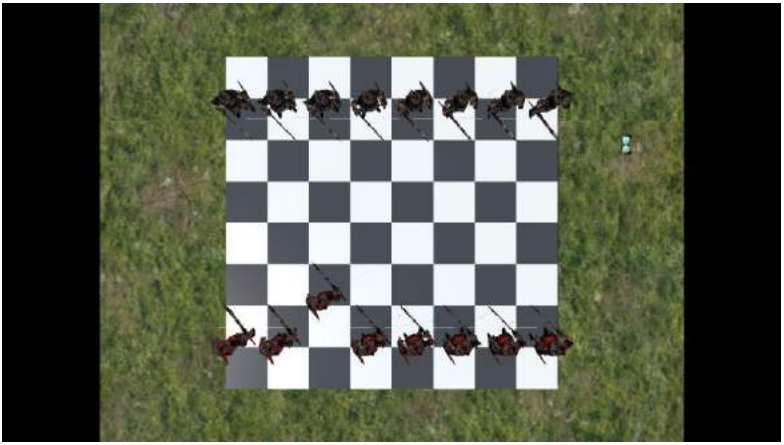
Skenario Pengujian hasil implementasi *rule-based strategy* dengan menanamkan kecerdasan buatan dengan level *hard* di salah satu tim pion dan menanamkan kecerdasan buatan dengan level *easy* di tim lain. Tujuan scenario ini adalah membuktikan bahwa kecerdasan buatan dengan level *hard* lebih pintar disbanding *easy*. Skenario ini tertera pada Tabel 5.3.

Tabel 5.3 Pengujian Permainan Antar Kecerdasan Buatan

	UFG01
Kondisi Awal	Masing-masing pion sudah menempati barisan masing-masing
Prosedur Pengujian	Permainan dimainkan menjadi 2 bagian. Bagian pertama, barisan pion putih ditanami kecerdasan buatan dengan level *hard* sedangkan barisan pion hitam ditanami dengan kecerdasan buatan dengan level *easy*. Bagian kedua, barisan pion putih ditanami kecerdasan buatan dengan level *easy* sedangkan barisan pion hitam ditanami dengan kecerdasan buatan dengan level *hard*.
Hasil yang diharapkan	Barisan pion dengan level *hard* memenangkan pertandingan.
Hasil yang diperoleh	Barisan pion *hard* memenangkan pertandingan ketika ditanamkan di barisan pion hitam maupun putih.
Kesimpulan.	Pengujian berhasil.

5.2.2.1 PAI01: Pengujian Level Hard Melawan Level Easy

Pengujian dimulai dengan barisan pion putih yang sudah ditanami kecerdasan buatan level *hard* melangkah terlebih dahulu.



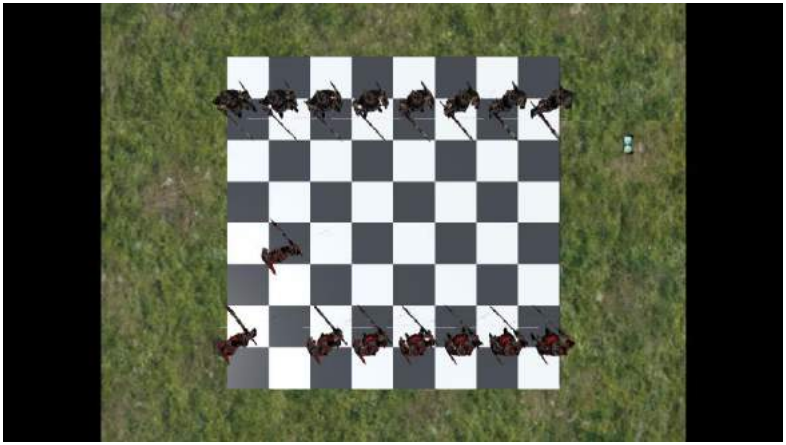
Gambar 5.12 Langkah pertama dalam pertandingan hard melawan easy

Setelah beberapa saat pertandingan berlangsung, akhirnya pertandingan dimenangkan oleh barisan pion putih yang sebelumnya telah ditanami dengan kecerdasan buatan level *hard*.

Gambar 5.13 Hasil akhir pertandingan hard melawan easy

5.2.2.2 PAI02: Pengujian Level Easy Melawan Level Hard

Pengujian dimulai dengan barisan pion putih yang sudah ditanami kecerdasan buatan level *easy* melangkah terlebih dahulu.



Gambar 5.14 Hasil akhir pertandingan hard melawan easy

Gambar 5.14 Hasil akhir pertandingan hard melawan easy

5.3 Pengujian Performa Kecepatan Berpikir NPC

Pengujian dilakukan untuk menguji seberapa cepat sistem menghitung semua langkah yang mungkin untuk tingkat kesulitan *easy*, *medium* dan *hard*.

5.3.1 Skenario dan Data Uji Coba Kecepatan.

Skenario Pengujian kecepatan performa digunakan untuk memberikan tahap-tahap dalam pengujian sistem dalam hal kecepatan. Skenario ini tertera pada Tabel 5.4.

Tabel 5.4 Skenario Pengujian Performa

Deskripsi	Bertujuan untuk mengetahui performa sistem dalam menghitung semua langkah yang mungkin untuk setiap tingkat kesulitan
Langkah pengujian	1. Memilih salah satu level dan menekan tombol klik. 2. Sistem menampilkan layar **Kondisi Awal Permainan**. 3. Menjalankan salah satu pion. 4. Ketika giliran selesai, sistem akan mulai menghitung semua kemungkinan langkah dari masing-masing pion dengan kedalaman tertentu 5. Sistem berhasil menemukan langkah terbaik dan memindahkan pion miliknya. 6. Pada *console* Unity muncul Debug.Log berisi kecepatan pembangkitan level permainan dalam detik, catat ke dalam tabel. 7. Ulangi langkah ke-1 sampai ke-6. Jika semua area sudah dicoba sebanyak lima kali, menuju langkah ke-8. 8. Cari rata-rata waktu kecepatan pembangkitan level.

Pengujian dilakukan dengan cara memilih satu persatu level yang ingin dibangkitkan pada saat layar **Pemilihan Tingkat Kesulitan**. Waktu berpikir NPC dalam satuan detik dicatat

setelah Debug.Log kerluar. kemudian kembali ke layar **Pemilihan Tingkat Kesulitan** dan memilih level yang lain.

5.3.2 Hasil Pengujian Performa

Hasil pengujian performa dikhususkan untuk menghitung waktu yang dibutuhkan NPC untuk berpikir sebagaimana dijelaskan pada Tabel 5.5. semua nilai pada tabel dihitung dalam satuan detik.

Tabel 5.5 Hasil Uji Coba Performa Waktu Berpikir NPC

level/Waktu dalam milidetik	Uji Coba ke-1	Uji Coba ke-2	Uji Coba ke-3	Uji Coba ke-4	Uji Coba ke-5	Rata-rata	Rata-Rata Tiap level
Easy langkah 1	0.13	0.1	0.12	0.1	0.1	0.110	0.084
Easy langkah 2	0.11	0	0.1	0.1	0.1	0.082	
Easy langkah 3	0.12	0.14	0	0.1	0.1	0.092	
Easy langkah 4	0	0.1	0.1	0	0.12	0.064	
Easy langkah 5	0.1	0	0	0.1	0.1	0.060	
Medium langkah 1	1.2	1.5	1.3	1.2	1.2	1.280	0.974
Medium langkah 2	1	1.1	1	1.15	0.98	1.046	
Medium langkah 3	0.92	0.96	0.9	1	1	0.956	
Medium langkah 4	0.91	0.89	0.8	0.9	0.81	0.862	
Medium langkah 5	0.72	0.73	0.73	0.71	0.75	0.728	
Hard langkah 1	4.2	4.1	4.1	4.1	4.1	4.120	3.636
Hard langkah 2	4	4	3.89	3.91	4	3.960	
Hard langkah 3	3.71	3.82	3.71	3.71	3.72	3.734	
Hard langkah 4	3.41	3.41	3.41	3.41	3.4	3.408	
Hard langkah 5	3	2.89	3	3	2.9	2.958	

Berdasarkan Tabel 5.5 dapat disimpulkan kecepatan berpikir NPC dengan level *easy* membutuhkan waktu rata-rata 0.084 detik, level *medium* membutuhkan waktu rata-rata 0.974 detik dan level *hard* membutuhkan waktu rata-rata 3.636 detik.

Waktu berpikir yang cukup singkat dibanding kecepatan berpikir manusia yang rata-rata membutuhkan 2 sampai dengan 4 detik.

5.4 Pengujian Pengguna

Pengujian pada perangkat lunak yang dibangun tidak hanya dilakukan pada fungsionalitas yang dimiliki, tetapi juga pada pengguna untuk mencoba secara langsung. Pengujian ini berfungsi sebagai pengujian subjektif yang bertujuan untuk mengetahui tingkat keberhasilan aplikasi yang dibangun dari sisi pengguna. Hal ini dapat dicapai dengan meminta penilaian dan tanggapan dari pengguna terhadap sejumlah aspek perangkat lunak yang ada.

5.4.1 Skenario Uji Coba Pengguna

Dalam melakukan pengujian perangkat lunak, pengguna diminta mencoba menggunakan perangkat lunak untuk mencoba semua fungsionalitas dan fitur yang ada. Serta untuk mencoba setiap tingkat kesulitan yang ada pada permainan.

Pengujian aplikasi oleh pengguna dilakukan dengan sebelumnya memberikan informasi seputar aplikasi, kegunaan, dan fitur-fitur yang dimiliki. Setelah informasi tersampaikan, pengguna kemudian diarahkan untuk langsung mencoba aplikasi dengan spesifikasi lingkungan yang sama dengan yang telah diuraikan pada uji coba fungsionalitas.

Jumlah pengguna yang terlibat dalam pengujian perangkat lunak sebanyak tiga orang. Dalam melakukan pengujian, pengguna melakukan percobaan lebih dari satu kali penggunaan untuk masing-masing pengguna.

Dalam memberikan penilaian dan tanggapan, penguji diberikan formulir pengujian perangkat lunak. Formulir pengujian perangkat lunak ini memiliki beberapa aspek penilaian dan pada bagian akhir terdapat saran untuk perbaikan fitur.

5.4.2 Daftar Penguji Perangkat Lunak

Pada subbab ini ditunjukkan daftar pengguna yang bertindak sebagai penguji coba aplikasi yang dibangun. Daftar nama penguji aplikasi ini dapat dilihat pada Tabel 5.6.

Tabel 5.6 Daftar Nama Penguji Coba Aplikasi

Nomor	Nama	Pekerjaan
1	Stieven Wirakasa	Mahasiswa Teknik Informatika ITS
2	Mulia Lovendo	Mahasiswa Teknik Informatika ITS
3	Alfian Maulana Azhari	Mahasiswa Teknik Informatika ITS
4	Rifi Febrio	Mahasiswa Teknik Informatika ITS
5	Andhik Ampuh Yunanto	Alumni Mahasiswa Teknik Informatika ITS

5.4.3 Hasil Uji Coba Pengguna

Uji coba yang dilakukan terhadap beberapa pengguna memiliki beberapa aspek yang dipisahkan berdasarkan antarmuka dan fungsionalitas yang dimiliki. Sistem penilaian didasarkan pada skala penghitungan satu sampai empat di mana skala satu menunjukkan nilai terendah dan skala empat menunjukkan skala tertinggi. Penilaian akhir kemudian dilakukan dengan menghitung berapa banyak penguji yang memilih suata skala tertentu dan kemudian dicari nilai rata-ratanya. Hasil uji coba dipaparkan secara lengkap dengan disertai tabel yang dapat dilihat pada subbab.

5.4.3.1 Hasil Penilaian Antarmuka

Penilaian antarmuka difokuskan pada penilaian pengguna terhadap kemudahan penggunaan antarmuka dan sifat-sifat lain yang perlu dimiliki. Hasil penilaian pengguna terhadap antarmuka aplikasi dapat dilihat pada Tabel 5.7.

Tabel 5.7 Penilaian Antarmuka

No.	Antarmuka	Penilaian				Rata-Rata
		1	2	3	4	
1	Kemudahan Penggunaan	0	0	1	4	3,8
2	Kelengkapan Menu	0	0	1	4	3,8
3	Keindahan Tampilan	0	2	1	2	3
4	Kecepatan Pemilihan Menu/Fitur	0	1	0	4	3,6
5	Kesesuaian tema	0	0	4	1	3,2
6	Ketertarikan Bermain	0	2	2	1	2,8
Nilai Akhir						**3,367**

5.4.3.2 Hasil Penilaian Performa Sistem

Penilaian performa sistem difokuskan pada penilaian pengguna terhadap kemampuan aplikasi dalam menghasilkan performa dari interaksi pengguna. Penilaian ini juga ditujukan untuk mendapatkan tingkat kecepatan dan kelancaran sistem atas interaksi yang dibuat oleh pengguna. Hasil penilaian performa sistem dapat dilihat pada Tabel 5.8.

Tabel 5.8 Penilaian Performa Sistem

No.	Performa Sistem	Penilaian				Rata-Rata
		1	2	3	4	
1	Performa atau kinerja pada permainan	0	0	1	4	3,8
2	Tingkat kepintaran NPC	0	1	2	2	3,2

3	Kecepatan berpikir NPC	0	0	2	3	3,6
4	Kelancaran animasi	0	1	2	2	3,2
5	Kesesuaian animasi dengan sistem permainan	0	0	2	3	3,6
6	Strategi bermain NPC	0	0	3	2	3,4
Nilai Akhir						**34,667**

5.4.4 Hasil Pengujian Pengguna

Evaluasi pengujian pengguna dilakukan dengan menampilkan data rekapitulasi perangkat lunak yang telah dipaparkan. Dalam hal ini, rekapitulasi disusun dalam bentuk tabel yang dapat dilihat pada Tabel 5.9. Dari data diketahui bahwa aplikasi telah memenuhi unsur yang seharusnya dimana nilai prosentase memiliki lebih dari 80%.

Tabel 5.9 Rekapitulasi Hasil Uji Coba Pengguna

No.	Nama Pengujian		Rata-Rata	Nilai	Nilai (%)
1	Penilaian Antarmuka	Kemudahan Penggunaan	3,8	**3,367**	**84,17**
		Kelengkapan Menu pada Permainan	3,8		
		Keindahan Tampilan	3		
		Kecepatan Pemilihan Menu/Fitur	3,6		
		Kesesuaian tema pada Permainan	3,2		
		Ketertarikan Bermain	2,8		

2	Performa Sistem permainan	Performa atau kinerja pada permainan	3,8	**3,467**	**86,675**
		Tingkat kepintaran NPC	3,2		
		Kecepatan berpikir NPC	3,6		
		Kelancaran animasi	3,2		
		Kesesuaian animasi dengan sistem permainan	3,6		
		Strategi bermain NPC	3,4		

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan perangkat lunak dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, terdapat pula saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1. Kesimpulan

Dalam proses pengerjaan tugas akhir mulai dari tahap analisis, desain, implementasi, hingga pengujian didapatkan kesimpulan sebagai berikut:

1. *Rule-based* berhasil diimplementasikan dengan bukti bisa mengalahkan lawan dalam beberapa pertandingan.
2. Kecerdasan buatan dalam permainan *battle pawn* berhasil dibuat dengan algoritma *alpha-beta pruning* sesuai dengan *rule-based* yang sudah ditentukan sebelumnya.
3. Berdasarkan uji performa, waktu berpikir NPC relatif naik secara eksponensial. Pada level *easy* dan *medium* NPC hanya membutuhkan waktu rata-rata kurang dari 1 detik. Namun ketika dihadapkan pada kedalaman algoritma berpikir yang lebih jauh, kenaikan waktu berpikir menjadi lebih lama hingga 3 kali lipat.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang, berdasarkan pada hasil perancangan, implementasi dan uji coba yang telah dilakukan.

1. Menambah lagi tingkat kesulitan agar pengguna bisa lebih bebas memilih lawan yang sesuai.
2. Menambah kondisi *heuristic value* agar NPC mampu berpikir lebih strategis.

DAFTAR PUSTAKA

[1] Unity, “Unity Game Development” 8 March 2015. [Online]. Available: http://www.unity3d.com/unity. [Accessed 2 June 2015].

[2] Blender Foundation, "Blender," 26 June 2014. [Online]. Available: http://www.blender.org/about. [Accessed 2 June 2015].

[3] Foraker Labs, "Usability Testing” March 2002. [Online]. Available: http://www.usabilityfirst.com/usability-methods/usability-testing/. [Accessed 3 June 2015].

[4] Goeller, Michael (2007) Pawn Battle Rules and Strategies.

[5] Wikipedia, “Wikipedia Alpha Beta Pruning”, 4 April 2015. [Online]. Available: http://en.wikipedia.org/wiki/Alpha-beta_pruning. [Accessed 3 June 2015]

[6] Russell, Stuart J.; Norvig, Peter (2010). Artificial Intelligence: A Modern Approach

[7] Blender Foundation, “Blender Supported Platforms”. 26 April 2015. [Online]. Available:http://wiki.blender.org/index.php/Dev:Ref/Supported_platforms. [Accessed 3 June 2015]

[8] Stack overflow “Currently known best algorithm(s) for chess”. 8 January 2010. [Online]: Available: http://stackoverflow.com/questions/2026262/currently-known- best-algorithms-for-computer-chess [Accessed 7 June 2015]

BIODATA PENULIS

Penulis dilahirkan di Surabaya, 25 Mei 1995, merupakan anak pertama dari 5 bersaudara. Penulis telah menempuh pendidikan formal yaitu TK Dharma Putra Surabaya (1998-2000), SD Negeri Gading 2 Surabaya (2000-2006), SMP Negeri 31 Surabaya (2006-2009), SMA IPIEMS Surabaya (2009-2012), dan mahasiswa S1 Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya rumpun mata kuliah Interaksi, Grafika, dan Seni (2012-2016).

Selama menjadi mahasiswa penulis pernah berperan sebagai asisten Kecerdasan Buatan (2014-2015). Penulis senang terlibat aktif dalam kegiatan sosial, beberapa kegiatan sosial yang pernah diikuti di antaranya adalah Bakti Sosial kampung temporejo mulyosari (2013-2014), kegiatan mitra KAMTIBMAS (2015) dan kegiatan-kegiatan sosial lainnya. Penulis yang memiliki hobi bermain *game* dan bereksplorasi diri. Penulis juga merupakan mahasiswa yang aktif dalam organisasi diantaranya himpunan mahasiswa Teknik Computer – Informatika ITS (HTMC) dan KMI ITS. Penulis dapat dihubungi melalui surel muhammadnear@gmail.com.